Deep Neural Networks to Solve Partial Differential Inverse Problems

Supervisors: Prof. Bangti Jin, Prof. Simon Arridge

Azhir Mahmood

September 2022

Abstract

In this thesis we demonstrate and compare multiple deep learning architectures performance at solving nonlinear inverse problems. The specific problem which we we use as our example is the inverse problem associated with diffuse optical tomography which involves reconstructing the optical absorption properties of an object within a domain acquired from diffused photons collected at the boundary of the domain. We demonstrated the performance of three deep learning architects; multilayered perceptrons, convolutional neural networks and Fourier neural operators. Showing that the Fourier neural-based architecture can solve the diffuse optical tomography inverse problem. The gauss-newton algorithm acts as our gold standard. The Fourier neural operator shows superior performance when noise-levels are low achieving an average SSIM score of 0.94 (even out performing the Gauss-Newton algorithm), though performance dramatically deteriorates with the introduction of noise. Deep learning architectures, once trained, have an inference time in the order of seconds; generating 2000 images under 15 seconds. While the Gauss-Newton algorithm takes over 15 seconds to reconstruct a single image. Overall we find that the Fourier neural operator can outperform multilayered perceptrons and convolutional neural networks as well as the Gauss-Newton algorithm under certain constraints.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisors Dr. Bangti Jin and Prof. Simon Arridge for their support throughout this year.

I would also like to deeply thank Dr Nargiza Djurabekova, Imraj Singh and Riccardo Barbano for their motivation, as well. Ultimately, I would also like to thank my family.

The work is supported by the EPSRC-funded UCL Centre for Doctoral Training in Intelligent, Integrated Imaging in Healthcare (i4health) (EP/S021930/1).

Contents

Chapter 1

Introduction

In this thesis, we attempt to assess how effective deep learning architectures are at solving ill-posed inverse problems. Specifically, those where the forward operator is modelled by a set of partial differential equations. In pursuit of the answer to this question, we show that Fourier neural operator architectures are particularly effective at reconstructing images from measurements acquired about the boundary of a domain.

First, let us discuss why this is a particularly valuable or exciting problem. The value lies in the intersection of ideas. The attempt to solve this problem represents centuries of mathematics intersecting with the more modern field of machine learning. Thus, the capacity to solve partial differential equations using machine learning represents a significant advancement in humanity's ability to observe and model physical phenomena.

To properly set the context of the research undertaken in this thesis, we must provide a short introduction to partial differential equations. A partial differential equation is any equation which involves functions and their partial derivatives, thus imposing a relation between the various partial derivatives of a multivariable function. Researching such equations occupies a large sector of pure mathematics and is ubiquitous in mathematically oriented scientific fields, such as physics and engineering. Forming our modern scientific understanding of physical phenomena such as sound, heat, diffusion, electrostatics, electrodynamics,

thermodynamics, fluid dynamics, elasticity, general relativity, and quantum mechanics.

As it stands, this problem is broad, so it is necessary to break it down into smaller, easier-to-answer questions. By focusing on an imaging modality, we can further narrow the scope of the problem whilst also determining the nature of the partial differential equation we attempt to solve.

Thus the scope of the thesis is narrowly focused on applications of deep learning architectures on diffuse optical tomography and solving the inverse problem associated with diffuse optical tomography. The inverse problem for the diffuse optical tomography problem consists of recovering the spatial distributions of the optical parameters μ_a (and μ_s) in the domain Ω from boundary measurements of transmitted light at the surface $\delta\Omega$.[1] In less technical language, we are focused on reconstructing the optical absorption properties of tissue within a domain acquired from diffused photons collected at the boundary of the domain.

By improving the speed and accuracy for which partial differential inverse problems can be solved, we suddenly allow new imaging modalities to become far more valuable and viable within the hospital.

Chapter 2

Background

2.1 Inverse Problems and Classical Techniques to Solve Inverse Problems

The study of the inverse problem began early in the 20th century and remains an active field today; some could argue that the act of attempting to reconstruct a signal from a set of observations is a story as old as time. Science itself involves attempting to interpret a ground truth from a set of measurements or reconstructing the original signal from one which has undergone a corruption process. More formally, we would say that the inverse problem refers to using the results of observations to infer the values of the parameters that characterise a system and to estimate data that are not directly observable.[2] Such problems, which we collectively call inverse problems, are ubiquitous in science and our daily lives. They arise within physics, biology, medicine, engineering, finance and computer science.

A prominent sub-field of inverse problems is its applications to imaging, with examples found within chemistry, biology and medicine. Imaging provides us with a visual representation of information. Over the past few decades, numerous imaging methods have been developed, but what sets many of these imaging modalities apart from conventional photography is that the raw data is not immediately interpretable as an image directly. Significant

processing is required before the acquired data can be viewed as an image.

We see applications of such imaging modalities across disciplines, from microscopy and medicine to astronomy. In all these applications, the recorded images are distorted or corrupted versions of the ideal image. The inverse problem consists of reconstructing the idealised image from the measured one. Many imaging problems, such as the reconstruction of medical images (computer tomography, magnetic resonance imaging, electrical impedance tomography, positron-emission tomography) and deblurring or denoising of microscopy and astronomy images, are all instances of inverse problems.

Specifically, in the case of medical imaging, we generally think of attempting to visualise the internal structure of an object from an indirect set of measurements acquired externally from the object. In most cases, an individual may find an array of sources and detectors surrounding part of their body. We then project photons that pass through and interact with tissue within the human body. By taking measurements from the surface of their body, we can then reconstruct an image of the internal structure of the individual.

More formally, we can define inverse problems as a set of problems where one attempts to reconstruct a parameter from a set of indirect observations; given an outcome of a process, the inverse problem attempts to understand its causation. The most basic of inverse problems can be mathematically formulated as estimating a set of parameters, $f \in X$ from a set of measurements or data $g \in Y$ where

$$g = \mathbf{A}(f) \tag{2.1}$$

The forward operator $\mathbf{A} : X \mapsto Y$ is a noiseless mapping between two function spaces – the solution space, X, to the data space, Y. Any data acquisition process has an associated uncertainty or noise term η. As a result, we write

$$g = \mathbf{A}(f) + \eta. \tag{2.2}$$

A majority of inverse problems are ill-posed, so small errors in data may lead to large errors in the model parameter, or we may find several possible model parameter values that are consistent with observations. To better understand what we mean by an ill-posed problem, it is first necessary to define what we mean by a well-posed problem.

We can trace the ideas of well-posed and ill-posed problems back to the 20th-century French mathematician Jacques Hadamard who believed that mathematical models for all physical phenomena should share a set of properties. He described problems as being well-posed if they met a set of conditions — which we outline below. If these conditions are not met, the problem is termed ill-posed. (The reader might find it interesting to note that even if a problem is well-posed, it may still be ill-conditioned, meaning that a small error in the initial data can result in much larger errors in the answers).

Definition 2.1.1 (Well-posed problem)**.** Given a set of data g and set of parameters f, both of which abide by the following expression

$$g = \mathbf{A}(f), \tag{$\dagger$}$$

where $\mathbf{A}$ represents a forward operator. A problem is said to be well-posed if it exhibits the following properties:

1. A solution exists. For any data g, there exist parameters f that satisfy †, which means there exists a model that fits the observed data.

2. The solution is unique. For every data-set g, if there are f_1 and f_2 parameters that satisfy †, then $f_1 = f_2$. This means that the model that fits the observed data is unique.

3. The solution is stable and depends continuously on initial conditions. A^{-1} is a continuous map, which means small changes in the observed data g will make small changes

in the estimated model parameters f.

The problem is said to be ill-posed if any of the three properties do not hold.

Over the centuries, numerous methodologies have been devised to solve inverse problems. We will investigate techniques to solve such inverse problems, but before venturing down this path we must first distinguish between two types of inverse problems; linear inverse problems and nonlinear inverse problems. In the case where the inverse problem is said to be linear, the forward operator A is linear. Otherwise, and most often the case, we find the inverse problem is nonlinear. (It is also worth noting that models cannot always be described by a finite number of parameters. In this scenario, we look for distributed parameters, and as such, the goal of the inverse problem is to retrieve one or several functions. We generally describe such inverse problems as inverse problems with infinite dimensions).

Within this thesis, we are particularly interested in scenarios where the forward operator A is a partial differential equation. This is a common scenario, and the example we will later be exploring is the case of diffuse optical tomography, where A is modelled by a photon-diffusion equation. Before investigating this, it is necessary first to lay the groundwork and briefly introduce how inverse problems (both linear and nonlinear) have been solved in the past.

Classical Techniques to Sole Linear Inverse Problems

A common technique is to solve the inverse problem by first re-framing the problem as solving a system of linear equations; $\mathbf{A}\boldsymbol{f} = \boldsymbol{g}$ where $\mathbf{A}$ is a known matrix and $\boldsymbol{g}$ is a known vector.

In the case of linear inverse problems, we can write the forward operator A as a matrix $\mathbf{A} \in \mathbf{R^{MN}}$. In the case when $M = N$ and the matrix $\mathbf{A}$ has a full rank, the solution of the linear inverse problem is unique, and we can thus find the model parameters by multiplying the matrix inverse $\mathbf{A^{-1}}$ with the data $\boldsymbol{g}$, which in this case is written as a vector.

In many cases, it may not be practical to invert $\mathbf{A}$. This is especially the case when the dimension of the problem becomes large, direct inversion of $\mathbf{A}$ is often impossible and even storage of a matrix representation of $\mathbf{A}$ becomes difficult in cases where the dimensions are very large. As a result, numerous iterative methods have been developed. So in the presence of noisy observed data $\boldsymbol{g}$, the problem can be expressed as an optimization problem

$$\min_{f} ||\boldsymbol{g} - \mathbf{A}\boldsymbol{f}||_2^2 + J(\boldsymbol{f}) \tag{2.3}$$

where $J(\,\cdot\,)$ incorporates the prior information. For example, the Tikhonov regularization is popularly used, where $J(\mathbf{f}) = ||\Gamma\mathbf{f}||_2^2$ and Γ represents the Tikhonov matrix (e.g. $\Gamma = \alpha\mathbf{I}$). To solve an ill-posed inverse problem, extra knowledge of the system is usually needed, which we refer to as prior information. In the case of linear inverse problems, it is common to use techniques such as gradient descent or other such iterative algorithms to solve the problem.

Classical Techniques to Solve Nonlinear Inverse Problems

In the case of nonlinear inverse problems, we heavily rely on iterative methods to solve our problem. Thus it is easier to re-frame equation 2.3 as equal to an objective function $\Psi(\boldsymbol{f})$ and for simplicity we will be ignoring any regularisation terms. We are trying to find the $\boldsymbol{f}$ that minimises

$$\Psi(\boldsymbol{f}) = \arg\min ||\mathbf{A}\boldsymbol{f} - \boldsymbol{g}||_2^2. \tag{2.4}$$

To understand how minimisation can be achieved through a number of iterative algorithms, let us take the Taylor expansion of the objective function. Such that

$$\Psi(\boldsymbol{f}) = \sum_{\alpha \geq 0} \frac{(\boldsymbol{f} - \boldsymbol{f}_k)^\alpha}{\alpha!} (D^\alpha \Psi)(\boldsymbol{f}_k), \tag{2.5}$$

as a result we obtain

$$\Psi(\boldsymbol{f}) \approx \Psi(\boldsymbol{f}_k) + (\boldsymbol{f} - \boldsymbol{f}_k)^\top D\Psi(\boldsymbol{f}_k) + \frac{1}{2!}(\boldsymbol{f} - \boldsymbol{f}_k)^\top \{D^2\Psi(\boldsymbol{f}_k)\}(\boldsymbol{f} - \boldsymbol{f}_k), \tag{2.6}$$

where $D\Psi(\boldsymbol{f}_k)$ is the gradient of Ψ evaluated at $\boldsymbol{f} = \boldsymbol{f}_k$ and $D^2\Psi(\boldsymbol{f}_k)$ is the Hessian matrix. The Hessian matrix can also be written as $D^2\Psi(\boldsymbol{f}_k) = \mathbf{H}\Psi(\boldsymbol{f}_k)$. (A brief introduction to the Jacobian can be found within the appendix).

The value of using the Taylor expansion to approximate the objective function $\Psi(\boldsymbol{f})$ becomes clear by understanding it as the intuition behind the Newton Method[3], which we will now introduce. When applying these iterative methods to nonlinear inverse problem it is necessary to determine the step length via a line-search step and specify a stopping criterion for the algorithm to be terminated. In both the Newton and Gauss-Newton algorithms provided, we specify a maximum number of iterations $k_{\max}$ as our stopping criteria, but of course the stopping criterion can be a range of conditions; i.e. minimum error.

In the case of the Newton Method where the iterative Newton scheme is give by

$$\boldsymbol{f}_{k+1} = \boldsymbol{f}_k - (\mathbf{H}\Psi(\boldsymbol{f}_k))^{-1}\nabla\Psi(\boldsymbol{f}_k), \tag{2.7}$$

we can described the Newton Algrothm as:

Algorithm 1 Newton Method

Choose an initial guess $\boldsymbol{f}_0$ and number of iterations value $k_{\max}$
$k \leftarrow 0$
while $k < k_{\max}$ **do**
  $\boldsymbol{p}_k \leftarrow -(\,\mathbf{H}\Psi(\boldsymbol{f}_k)\,)^{-1}\,\nabla\Psi(\boldsymbol{f}_k)$
  $\tau_k \leftarrow \arg\min_{\tau>0}\Psi(\boldsymbol{f}_k + \tau\boldsymbol{p}_k)$
  $\boldsymbol{f}_k \leftarrow \boldsymbol{f}_k + \tau\boldsymbol{p}_k$
  $k \leftarrow k + 1$
end while

Notice how the above minimisation algorithm, the Newton method, make uses of both the Hessian and Gradient of the objective function. In some instances calculating the Hessian

provides no real challenge and the Newton method would be appropriate. This is particularly the case in the linear least-square problem $\Psi(\boldsymbol{f}) = ||\mathbf{A}\boldsymbol{f} - \boldsymbol{g}||^2$. Here the Gradient coincides with the negative residual associated with the least-squares criterion and is given by

$$-\nabla\Psi(\boldsymbol{f}) = \mathbf{A}^\top(\boldsymbol{g} - \mathbf{A}\boldsymbol{f}), \tag{2.8}$$

while the Hessian is given by

$$\mathbf{H}(\Psi(\boldsymbol{f})) = \mathbf{A}^\top\mathbf{A}. \tag{2.9}$$

We can see that equation 2.7 collapses down into solving the Moore-Penrose inverse $\mathbf{A}^\top A\boldsymbol{f}_1 = \mathbf{A}^\top\boldsymbol{g}$ in the case when $\boldsymbol{f}_0 = \mathbf{0}$

Though when we encounter a non-linear least-squared problem we can no longer rely on the Newton Method. It may no longer be feasible or possible to calculate the Hessian. An example of this is where the objective function being minimised could be of the form $\Psi(\boldsymbol{f}) = \frac{1}{2}||A(\boldsymbol{f}) - \boldsymbol{g}||^2$, where A is a non-linear operator our Hessian and Gradient of the objective function changes, such that the Gradient is given by $-\nabla\Psi(\boldsymbol{f}) = A^*(\boldsymbol{f})(\boldsymbol{g} - A\boldsymbol{f})$ and Hessian by $\mathbf{H}(\Psi(\boldsymbol{f})) = \nabla A^*(\boldsymbol{f})\nabla A(\boldsymbol{f}) - \mathbf{H}A(\boldsymbol{f})[\boldsymbol{g} - A(\boldsymbol{f})]$.

Notice that this is particularly difficult to calculate[4] and thus we introduce the Gauss-Newton algorithm[5] which uses the Jacobian to approximate the Hessian operator such that $\mathbf{H} = D^\top D$. This iterative algorithm is known as the Gauss-Newton; (Later in the thesis we will see it be used within our experiments acting as the gold standard). The changes mentioned are showcased below, within algorithm 2.

Algorithm 2 Gauss-Newton Method

Choose an initial guess f_0 and number of iterations value $k_{\max}$
$k \leftarrow 0$
while $k < k_{\max}$ **do**
  $\boldsymbol{p}_k \leftarrow -\{[D^\top D](\Psi(\boldsymbol{f}_k))\}^{-1} \nabla\Psi(\boldsymbol{f}_k)$
  $\tau_k \leftarrow \arg\min_{\tau>0} \Psi(\boldsymbol{f}_k + \tau\boldsymbol{p}_k)$
  $\boldsymbol{f}_k \leftarrow \boldsymbol{f}_k + \tau\boldsymbol{p}_k$
  $k \leftarrow k + 1$
end while

2.2 Deep Learning and their Application to Inverse Problems

One can argue that modern deep learning can trace its roots back to McCulloch and Pitt's 1943 paper which showed that simple elements connected in a neural network can have immense computational power.[6] The first machine that was able to implement this single-layer perceptron was built by Frank Rosenblatt in 1958 and arguably marks the the modern architectures of deep learn that we see today.[7] In recent years. More recently in the 2010s, advances in both machine learning algorithms and computer hardware have led to more efficient methods for training deep neural networks that contain many layers of non-linear hidden units and a very large output layer.[8] By 2019 graphical processing units (GPUs) had displaced central processing units (CPUs) as the dominant method of training large-scale machine learning algorithms in most commercial settings. Many have christened the era which we are witnessing today as the Deep Learning Revolution.[9].

One may wonder if classical techniques are effective what is the purposed of deep learning techniques applied to inverse problems? Well, earlier we saw how classical techniques can be employed to solve inverse problems, though they are effective they are computationally expensive. In contrast with iterative techniques, more recently deep learning has given rise to a range of new techniques being employed to solve partial differential equations and inverse

problems.

In general if we recall how inverse problems are defined by a mapping between two spaces; a data space and a solution space. We can use a neural network to learn this mapping. As early as 1989 Kurt Hornik established that multilayer feedforward networks are a class of universal approximators, thus The Universal Approximation Theorem explicitly states that a neural network with 1 hidden layer can approximate any continuous function for inputs within a specific range.[10]. Though in the case of partial differential equations the mapping between data space and solution space is not given by a function, but in fact an operator. In 1995 Tianping Chen and Robert Chen were able to expand upon the Universal Approximation Theorem and show that that a single hidden layer can accurately approximate any nonlinear continuous operator.[11] This was made much more explicit in 2020 by George Em Karniadakis et al. in what was christened as the Universal Approximation Theorem for Operators.[12]

In short, there has been a lot of recent work which shows it is possible to learn mappings between function spaces, this acceleration within the deep learning community acts as our motivation for why the use of deep learning architectures to solve non-linear inverse partial different equations is appropriate. Since

In this thesis we focus on supervised learning techniques, which require training-data, heavily focusing in on architecture such as the Multilayer Perceptron (MLP)[13], Convolutional Neural Network (CNN)[14] and the Fourier Neural Operator (FNO)[15].

2.2.1 Fully Connected Networks

A Fully Connected Neural Network (FCN), also referred to as a multilayer perceptron (MLP), consists of three or more layers (an input and an output layer with one or more hidden layers between them).[7,6] Each layer is composed of a set of nodes, what makes a layer fully connected is that every node from the previous layer is connected to every nodes in the current layer. Figure 2.1 provides a visual representation of the structure of the network.

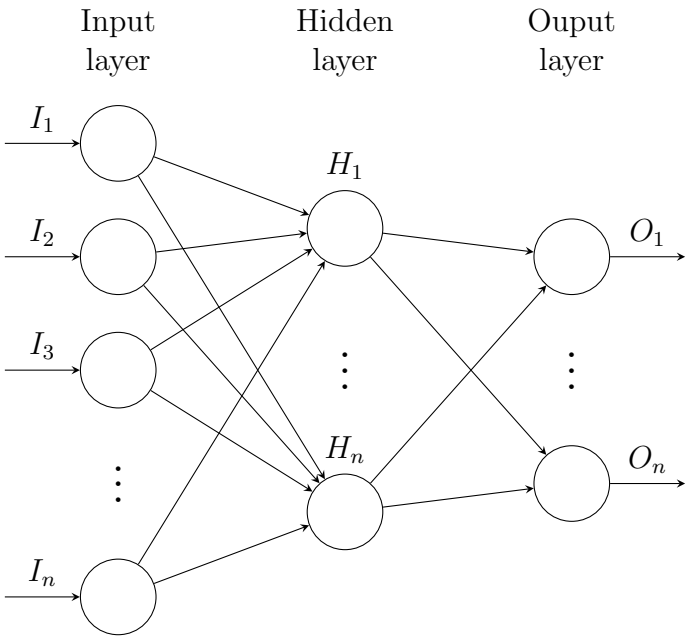

Figure 2.1: A fully connected neural network, also called a multilayer perceptron. This network only has a single hidden layer. Note that I_n, H_n and O_n represent input, hidden and output nodes.

The general algebraic representation of a general single hidden-layer unit is a linear combination of inputs passed through a nonlinear 'activation' function σ.[16] Between hidden layers we use ReLu as the activation function. Explicitly, at each node the input is multiplied by a weight $\mathbf{W}$ and has a bias $\mathbf{b}$ added to it, before being passed through the activation function σ and sent over to the next set of nodes.

In the notation of inverse problems our input is $g(\mathbf{x})$ and the reconstruction / output of the network is $f^{\#}(\mathbf{x})$. Within a multilayer perceptron, data is vectorised and as such $f^{\#}(\mathbf{x})$ becomes $\mathbf{f}_i^{\#}$, while $f(\mathbf{x})$ becomes $\mathbf{f}_i$. Our measurements $g(\mathbf{x})$ must also be vectorised and thus the input into our network is $\mathbf{h}_i^{[0]} = \mathbf{g}_i$ with corresponding ground truths $\mathbf{f}_i$ — in supervised learning a network provided training data $\{\mathbf{g}_i, \mathbf{f}_i\}_{i=1}^m$. For a general multilayer perceptron with N layers, the output from layer n is

$$\mathbf{h}_k^{[n]} = \sigma^{[n]}\Big(\mathbf{W}_{ki}^{[n]}\ \mathbf{h}_i^{[n-1]} + \mathbf{b}_k^{[n]}\Big). \tag{2.10}$$

The output of the final layer N being $\mathbf{h}_i^{[N]} = \mathbf{f}_i^{\#}$. Generally we train our network to minimise

the error between the reconstruction $f^{\#}(\mathbf{x})$ and our ground truth $f(\mathbf{x})$. With the correct weights $\mathbf{W}_{ij}$ and biases $\mathbf{b}_i$, a multilayer perceptron architecture can be used to approximate any function.

This property is explicitly given by the Universal Approximation Theorem, which states that a neural network with 1 hidden layer can approximate any continuous function for inputs within a specific range.[10]. This can be expanded upon and it is shown that a single hidden layer can accurately approximate any nonlinear continuous operator.[12] This is the motivation behind using deep learning architectures to solve non-linear inverse partial different equations.

2.2.2 Convolutional Neural Networks

A Convolutional Neural Networks (CNN) is able to successfully capture the spatial and temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image data-set due to the reduction in the number of parameters involved and re-usability of weights. In other words, the network can be trained to better understand the sophistication of the image.[17,18]

Convolutional Neural Networks use relatively little pre-processing compared to other image classification algorithms. This means that the network learns to optimize the filters (or kernels) through automated learning, whereas in traditional algorithms these filters are hand-engineered. This independence from prior knowledge and human intervention in feature extraction is a major advantage.[19]

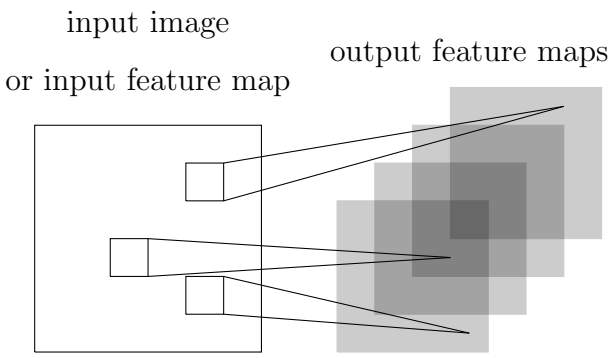

Figure 2.2: Illustration of a single convolutional layer. If layer l is a convolutional layer, the input image (if $l = 1$) or a feature map of the previous layer is convolved by different filters to yield the output feature maps of layer l.

The convolutional layer is the core building block of a Convolutional Neural Network. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, producing an activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input.

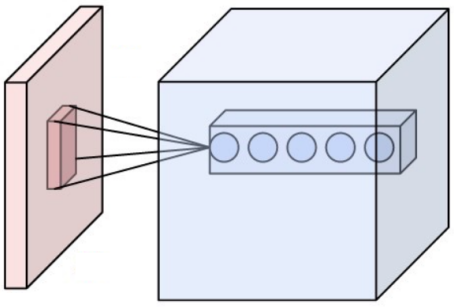

Figure 2.3: The input being projected to its feature map by a convolution layer

More explicitly when a convolutional layer acts on an image, which we represent as a tensor with a shape (number of inputs) × (input height) × (input width) × (input channels), the image becomes abstracted to a feature map, also called an activation map, with shape: (number of inputs) × (feature map height) × (feature map width) × (feature map channels).

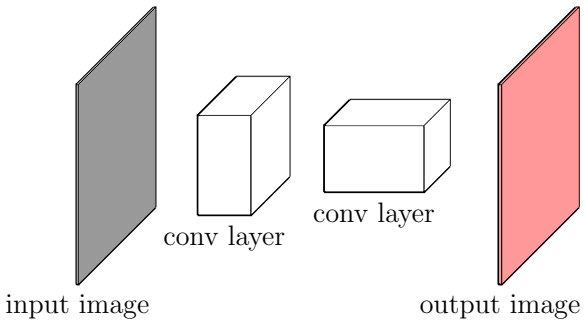

Figure 2.4: An example of a convolutional neural network architecture. The input image, which is indicated in grey, is processed by two convolutional layers

We can stack multiple convolution layers together to extract increasingly more complex features from an image, as seen in figure 2.4. The first convolutional layer is responsible for capturing the low level features such as edges, color, gradient orientation, etc. Each successive convolutional layer capture increasingly higher level feature.

2.2.3 Fourier Neural Operators

In 2021 Andrew Stuart et al.[20] proposed architectures a class of architectures which act as generalisations of neural networks. Their claim being that while neural networks focused on learning mappings between finite dimensional Euclidean spaces or finite sets, their architecture is able to learn operators that map between infinite dimensional function spaces. They collectively named these class of class of architectures as neural operators; which the Fourier neural operator falls under.

Neural operators are generalisations of the standard feed forward neural networks. Allowing them to learn mappings between infinite-dimensional spaces of functions defined on bounded domains. Non-local components are learnt via a parameterised integral operator, as is the case of the graph-based operators,[21] or through multiplications within the spectral domain, as seen with the Fourier neural operator (FNO).

The architecture of the Neural operators are comprised of four step:

1. Extracting features from the input functions.

2. Iterating a recurrent neural network on the feature space.

3. Finally, mapping from feature space into the output function.

Now having covered the general structure of neural operators, we can return to the Fourier neural operator. Similar to our earlier neural network architectures discussed, the Fourier neural operator is a supervised learning method that attempts to learn the operator between two infinite dimensional spaces from a finite data set. In the case of partial different equation's this is understood as the solution operator, which is a non-linear map $\mathbf{M}^{\dagger} : Y \mapsto X$, where $f_j = \mathbf{M}^{\dagger}(g_j)$. (Note that f_j exists within the space X, while g_j exists within the space Y, and both exist within the real space $\mathbb{R}$).

Given a data set $\{f_j, g_j\}_{j=1}^{N}$, the solution operator $M^{\dagger}$ is approximated by constructing a parametric mapping such that

$$M_{\theta} : Y \mapsto X, \quad \theta \in \Theta \tag{2.11}$$

for some finite-dimensional parameter space Θ by choosing $\theta^{\dagger} \in \Theta$ so that $G(\cdot, \theta^{\dagger}) = G_{\theta}^{\dagger} \approx G^{\dagger}$.

By construction the method shares the same learned network parameters irrespective of the discretization used on the input and output spaces, allowing it to perform zero-shot super-resolution: trained on a lower resolution directly evaluated on a higher resolution.[15]

Motivation

Given a linear differential operator $\mathcal{L} = \mathcal{L}(x)$ acting on distributions over a subset of the Euclidean space $\mathbb{R}^n$, a Green's function $G_s = G(x, s)$ at the point s is any solution of

$$\mathcal{L}\mathcal{G}_s(x) = \delta(x - s). \tag{2.12}$$

In essence, a differential operator is a linear combination of derivative operators times functions and the Green's function is an inverse of an arbitrary linear differential operator.

Definition 2.2.1 (Linear Differential Operator)**.** A Differential Operator is a mapping $\mathcal{L}$ from one function space F_1 to another function space F_2. An example of a linear differential operator is

$$\mathcal{L} = \frac{\partial^2}{\partial x^2} + \frac{\partial}{\partial x \partial y} + \frac{\partial}{\partial y \partial x} + \frac{\partial^2}{\partial y^2} + \frac{\partial}{\partial x} + \frac{\partial}{\partial y} \ .$$

in this scenario our linear differential operator contains partial derivatives. In general an operator is said to be linear if

$$\mathcal{L}(a_0(x) + a_1(x)) = \mathcal{L}(a_0(x)) + \mathcal{L}(a_1(x))$$

and

$$\mathcal{L}(c\ a_0(x)) = c\ \mathcal{L}(a_0(x)),$$

If we consider a general linear partial different equation with constant coefficients,

$$\mathcal{L}f(x) = g(x), \tag{2.13}$$

where $\mathcal{L}$ is the differential operator. (Note, that this does not hold for differential operators with non-constant coefficients). We are also assuming that the domain is infinite — the reason why we can make this assumption is that a heavily discretised domain may locally appear infinite, thus we are also ignoring boundary conditions for the time being. Now if we take the Fourier transform of our expression, we obtain

$$\widetilde{\mathcal{L}}(k)\widetilde{f}(k) = \widetilde{g}(k). \tag{2.14}$$

Considering the Fourier transforms differentiation properties, $\mathscr{F}\{\frac{d^n f(x)}{dx^n}\} = (ik)^n\ \widetilde{f}(k)$, it is evident that $\widetilde{\mathcal{L}}(k)$ is a simple polynomial which we can manipulate and divide by.

Separately, the Green's function $\mathcal{G}$ is the solution of the equation

$$\mathcal{L}\mathcal{G}_s(x) = \delta(x - s), \tag{2.15}$$

where δ is Dirac's delta function. Taking the Fourier transform of the above expression results in,

$$\widetilde{\mathcal{L}}(k)\ \widetilde{\mathcal{G}}_s(k) = 1 \tag{2.16}$$

$$\widetilde{\mathcal{G}}_s(k) = \left(\widetilde{\mathcal{L}}(k)\right)^{-1}. \tag{2.17}$$

As mentioned earlier, in the above expression we are able to divided through $\widetilde{\mathcal{L}}(k)$, thus allowing us to substitute 2.17 into 2.17 obtaining

$$\widetilde{f}(k) = \widetilde{\mathcal{G}}_s(k)\ \widetilde{g}(k) \tag{2.18}$$

$$f(x) = \mathscr{F}^{-1}\{\widetilde{\mathcal{G}}_s(k)\ \widetilde{g}(k)\} \tag{2.19}$$

By the convolution theorem we know that

$$f(x) = \mathcal{G}(x) * g(x), \tag{2.20}$$

is equivalent to

$$f(x) = \int_{-\infty}^{\infty} \mathcal{G}(x - s)\ g(s)\ ds \tag{2.21}$$

The above derivation can be expanded for multiple dimensions, though gives an intuition behind why taking the Fourier transform can be so valuable when solving partial different equations. The Fourier transform is integral to solving a partial different equation using Green's function and necessary for understanding the mechanism in which Green's function

can be used to solve such partial different equation's. Thus it is not surprising that by taking the Fourier transform of a data set it greatly improves the ability of a neural network to learn an operator.

Architecture and Theoretical Framework

Having now understood the motivation behind the architecture, we can investigate it in greater depth. By parameterising the integral kernel directly in the Fourier space, they created the first ML-based model capable of successfully modelling turbulent flow with zero-shot super-resolution; performing three orders of magnitude faster than traditional partial different equation solvers.

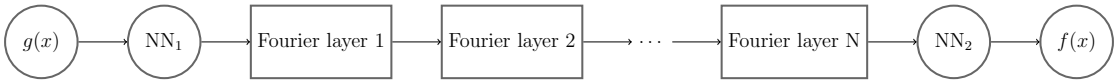

Figure 2.5: The full architecture of neural operator: starting from input $g(x)$, we lift our dataset to a higher dimensional space by a neural network NN_1; this in essence is identical to increasing the number of channels of our data-set. Then apply four layers of integral operators and activation functions; these are our fourier layers. Finally projecting it back to the target dimension by a neural network NN_2; which is identical to returning to the original number of channels. Giving us our output $f(x)$.

In figure 2.5 we can observe the full architecture of the Fourier neural operator. By increasing the number of channels of $g(x)$ through a fully-connected network NN_1, we increase the number of learnable parameters. Then apply four layers of integral operators and activation functions; these are our fourier layers. Finally returning to the original number of channels by a neural network NN_2, giving us our output $f(x)$. (The operation of increasing and reducing the number of channels is referred to as projecting into a higher dimensional space and projecting to the target dimensional space).

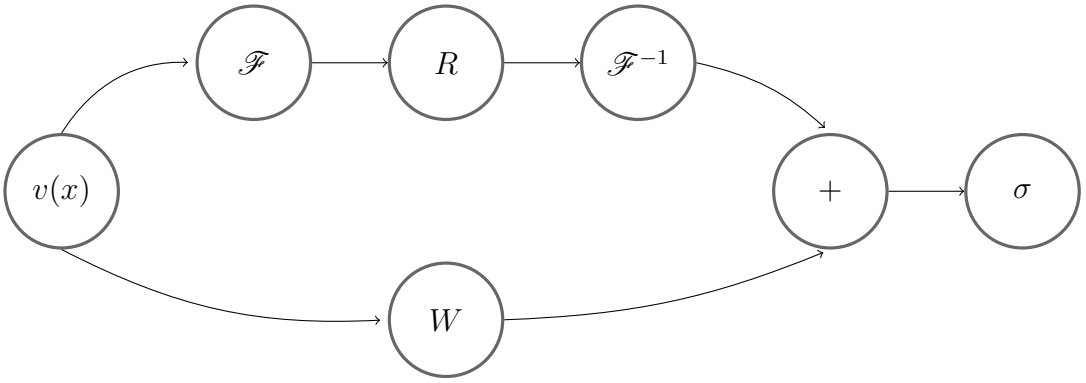

Figure 2.6: Fourier layers: Our input dataset is higher dimensional plane is in a higher dimensional plane, thus denoted as $v(x)$. Top: After applying a fourier transform $\mathcal{F}$, we filtered out the the high frequency modes using a low pass filter R. Before then apply the inverse Fourier transform on the remainder $\mathcal{F}^1$. Bottom: apply a local linear transform W. R is known as a spectral filter and W is a standard convolutional filter.

We can summarise the architecture as

$$v^{[0]} = \mathrm{NN}_1(\, g(x)\,) \tag{2.22}$$

$$v^{[n+1]} = \sigma(\, \mathscr{F}^{-1}\{R\mathscr{F}\{v^{[n]}\}\} + Wv^{[n]}\,) \tag{2.23}$$

$$f(x) = \mathrm{NN}_2(\, v^{[N]}\,). \tag{2.24}$$

The architecture projects the data $g(\mathbf{x})$ point-wise into a higher dimensional plane. That is to say a data-set with dimensions $8 \times 16 \times 1$ is projected by NN_1 as a representation of itself $v(\mathbf{x})$ that is $8 \times 16 \times d$ where d here is the depth of the representation which can be thought of as the number of channels — as is the case with convolutional layers. Once various Fourier layers are applied to the representation of the input image $v(\mathbf{x})$, it projected back into its original dimension. That is to say $v(\mathbf{x})$ is flattened or reshaped from $8 \times 16 \times d$ to the desired image size which in our case is $16 \times 16 \times 1$. More generally in the architecture, NN_1 is fully connected layers which take in the data $g(x)$ through a single node and project it to multiple nodes; thus it acts as a one to many mapping. After the various Fourier layers NN_2 is a second fully connected layer which acts as a send mapping to project $v(\mathbf{x})^{[N]}$ into

the solution space — acting as a many to one mapping. (It is also worth noting that R is known as a spectral filter and W is a standard convolutional filter).

2.2.4 Network Training

In practice the network must learn the correct weights and biases. In the case of supervised learning, this is achieved through a 'feed-forward' process that involves feeding the data forward, through the layers in the network, and making prediction. Then we calculate the error between the ground truth and the predictions, before finally propagating the error backwards, through the layers, by updating the weights and biases within the network. The error of the network is calculated through an objective function (which is generally referred to as a lost function) $\Psi(\Theta)$, where Θ represents the parameters of the network. $\Psi(\Theta)$ is usually a collection of the various the errors within each node, these errors are calculated using either mean-squared error or a similar cost function.

A common method by which a neural network architecture is trained is via gradient descent. It was independently discovered by Augustin-Louis Cauchy in 1847[22] and Jacques Hadamard in 1907[23].

Algorithm 3 Gradient Descent

After the forwards computation is completed, we begin our back propagation. Going backwards through our network, we update our weights and biases based on an objective function Ψ. More advanced loss functions also update the learning rate α.

Require:
Output $\mathbf{y}^{\#}$ from our feed-forward and the expected results $\mathbf{y}$ which we evaluate against.

$\Psi \leftarrow \Psi(\mathbf{y}^{\#}, \mathbf{y})$ ▷ Input ground truth and reconstruction into loss function.
for $k = l, l-1, .., 1$ **do**
  $\mathbf{W}^{[k]} \leftarrow \mathbf{W}^{[k]} - \alpha \frac{\partial \Psi}{\partial \mathbf{W}^{[k]}}$
  $\mathbf{b}^{[k]} \leftarrow \mathbf{b}^{[k]} - \alpha \frac{\partial \Psi}{\partial \mathbf{b}^{[k]}}$
end for
  return Update weights $\mathbf{W}^{[k]}$ and biases $\mathbf{b}^{[k]}$ across network.

Mini-batch gradient descent has become the most common implementation of gradient descent used in the field of deep learning. To improve the stability of a optimisation algorithm data is usually split into smaller batches that are then used to calculate model error and update model coefficients. Implementations may choose to sum the gradient over the mini-batch which further reduces the variance of the gradient. This in turn finds a balance between the robustness of methods that calculated error for every data-point and the efficiency of calculating error across the whole data-set.

Algorithm 4 Mini-Batch Gradient Descent

We initialise the algorithm within the network by randomly assigning values to all the weights $\mathbf{W}_{ik}$ and biases $\mathbf{b}_k$. The network then attempts to minimise the error between the reconstruction $\mathbf{f}_i^{\#}$ it outputs and the ground truth $\mathbf{f}_i$. We first split our training data-set $\{\mathbf{g}_i, \mathbf{f}_i\}_{i=1}^m$ into b batches of equal size m' and thus create subsets of training-data $\{\mathbf{x}_i^{[k]}, \mathbf{y}_i^{[k]}\}$, where $k \in \{1, .., b\}$ corresponds to batch numbers and $i \in \{1, ..., m'\}$ corresponds to data-set within batch.

Require:
Mini-batched data-set $\{\mathbf{x}_i^{[k]}, \mathbf{y}_i^{[k]}\}$.

We selected a batch of data to loop through and compare the reconstructions given our current parameters $\Theta \in \{\mathbf{W}_{ij}\mathbf{b}_j\}$, collate the errors, update our parameters before moving to the next batch. We specify the number of epoch the networks has been trained over as j, which is initialised as $j = 1$.

while not converged **do**
  for $k = 1, .., b$ **do**
    $\Psi = \frac{1}{2m'} \sum_{i=1}^{m'} (\mathbf{h}_\Theta(\mathbf{x}_i^{[k]}) - \mathbf{y}_i^{[k]})^2$ ▷ Our objective function is MSE
    $\Theta_{k+1,j} \leftarrow \Theta_{k,j} - \alpha \cdot \nabla_\Theta \Psi$
  end for
  $j \leftarrow j + 1$
  Test for convergence
end while
Output: Set weights $\mathbf{W}$ and bias $\mathbf{b}$ of network from Θ.

After a data-set is passed through the network either batch-wise or in its entirety, the

network uses an objective function to improve accuracy. The objective function takes in the current output of the network and compares it to the expected values, if the error is above a specified criterion, the objective function updates the weights and biases within the network to improve accuracy.

The weights of the a network are updated via a process known as backpropogation, which works by computing the gradient of the loss function with respect to each weight by the chain rule, computing the gradient one layer at a time, iterating backward from the last layer to avoid redundant calculations of intermediate terms in the chain rule.[24]

In most cases the cost function is the means-square-error between the network output and ground truth, while the objective function is the sum of the costs. The usual optimization algorithm used is the Adam optimiser[25], which updates the learning rate α alongside the weights $\mathbf{W}$ and biases $\mathbf{b}$.

2.3 Diffuse Optical Tomography

Within this thesis, the imaging modality which we use as an example problem is diffuse optical tomography (DOT). As mentioned earlier the work in the thesis is narrowly focused on applications of deep learning architectures on diffuse optical tomography and solving the inverse problem associated. We are attempting to assess the performance of deep learning architectures at recovering the spatial distributions of the optical parameters μ_a in the domain Ω from boundary measurements of light collected at the surface $\delta\Omega$. This problem is nonlinear and severely ill-posed. It is worth our time providing a short introduction to diffuse optical tomography.

There has been some success in using deep learning methods to solve the inverse diffuse optical tomography problem. Though overall, the use of deep learning methods remain scarce due to difficulty to obtaining high quality training data, costly model evaluations, and limitations of direct reconstruction approaches.[26] I.e. In 2020 Seungryong Cho et al. showed

the convolutional neural networks based architectures were effective at estimating bulk optical properties in diffuse optical tomography which out performed existing least squared methods in terms of accuracy and computation time.[27] While in 2021 Andreas Hauptmann et al. showed that a model-based iterative learning approach, where learned components are intertwined with the model equation, to be effective and was able to compensate for modelling errors.[28]

It is a non-invasive imaging modality that reconstructs a volumetric map of an object by measuring the scattering and absorption of light through the object. This medical imaging modality is particularly valuable for imaging soft tissue has been applied in various deep-tissue applications such as breast cancer imaging, brain functional imaging and neuroimaging.[29]

Within diffuse optical tomography, tissue is illuminated with visible or near-infrared light at at a wavelength of 650 — 1000 nm. The imaging modality takes advantage of the relative 'transparency' of biological tissue in the near infrared that allows light to penetrate several centimeters. The photon beams are projected across the object in parallel beams to an array of photo-detectors. The process is repeated at various angles, illuminating the bodies surface at different source locations in succession, allowing us to reconstruct a 3D image of the object being measured.

A strength of this imaging modality is that red and infrared wavelengths can easily pass through structures such as the skull and brain, as well as the body capacity tolerated large dosages of such waves. Overall the instrumentation is noninvasive, non-ionizing, inexpensive and can be made portable. It has the potential for real-time continuous data acquisition at relatively large penetration depth.[30]

Though given biological tissue is a highly scattering medium the photons take very irregular paths. There are several molecules that have characteristic absorption spectra. In particular, the spectra of oxyhemoglobin, deoxyhemoglobin, and cytochrome oxidase differ considerably. Hemoglobin provides an indicator of blood volume and its oxygenation, whereas the

cytochrome enzymes indicate tissue oxygenation (which can be called absorbers).[31] Between 650 — 1000 nm they have relatively weak absorption, thus providing a spectral window through which we can attempt to localize absorption. (At frequencies higher absorption by water increases rapidly).[32]

The Beer–Lambert law states that there is a linear relationship between the concentration and the absorbance of the solution.[33,34] Which allows us to determine the tissue oxygen saturation and haemoglobin content via the difference in intensity between a transmitted and received light delivered at specific wavelengths. The depth of penetration is proportional to the mean path length of photons travel through the tissue.

The transmission of light at a given wavelength through tissue is dependent on a combination of factors including reflectance, scattering, and absorptive effects. The impact of reflectance decreases at increasing wavelengths, favouring the use of near-infrared light over visible light. Scattering on the other hand is influenced by of tissue composition and number of tissue interfaces, while absorption is determined by the molecular properties of substances within the light path.

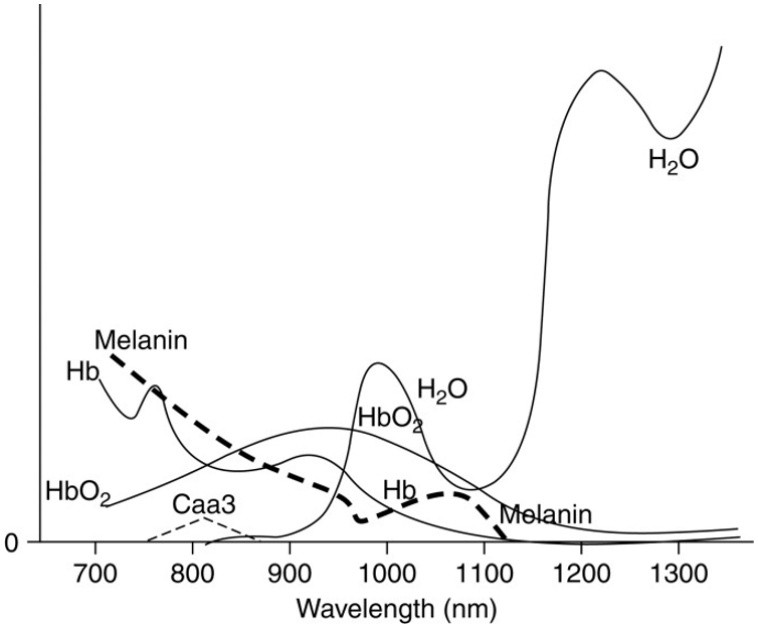

Figure 2.7: Absorption spectra for oxygenated haemoglobin (HbO_2), deoxygenated haemoglobin (Hb), Caa3, melanin, and water over wavelengths in near-infrared range.[35]

Figure 2.7 provides the reader with a graph which explains the absorption characteristics of different molecules found within the body which influence the effectiveness of diffuse optical tomography. Above 1300 nm, water absorbs all photons over a path length of a few millimetres with a secondary peak between 950 and 1050 nm. While below 700 nm, increased light scattering and intense absorption bands of haemoglobin prevent effective transmission.

Thus by using near-infrared light at a wavelength between 700–1300 nm, it is possible to penetrates several centimetres biological tissue. Within this range the primary light-absorbing molecules in tissue are metal complex chromophores: haemoglobin, bilirubin, and cytochrome. In commercial devices the wavelength of near-infrared light utilised is between 700 and 850 nm, in order to be sensitive to biologically important chromophores. Maximising the separation between absorption spectra of oxygenated haemoglobin (HbO_2) and deoxygenated haemoglobin (Hb) , as well as ensuring there is minimal overlap with the absoprtion spectra of water.[35,36]

In essence diffuse optical tomography offers the opportunity to image 3-D spatial variations in blood parameters, particularly hemoglobin concentration and oxygen saturation, and thus metabolic factors which these concentrations reflect, along with tissue scattering characteristics.

2.3.1 Experimental Protocol for Diffuse Optical Tomography

For completeness, we have included a short explanation of the experimental protocols involved when using this imaging modality. Hopefully it will act as a guide to provide the reader with an intuition of how diffuse optical tomography functions in a practical setting. As an imaging modality there are numerous practical considerations, ranging from the arrangement of detectors to the measurement protocol.

The geometries chosen for many laboratory and clinical near-infrared imaging studies fall into three general categories; projection-shadow, circular-tomography and subsurface imaging. It should be noted that the projection-shadow geometry is generally used for traditional

x-ray film style images, where there is no inversion of the data to compute tomographic images, and that this geometry does not lend itself to tomographic reconstruction.[37] (In figures 2.8 to 2.10 the blue arrows represent detectors, while red arrows represent sources).

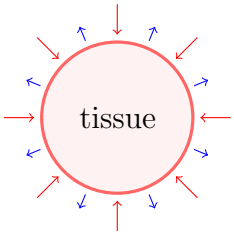

Figure 2.8: Circular tomography, usually used in breast imaging. Ideal for 3D reconstruction.

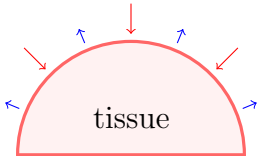

Figure 2.9: Sub-surface imaging, commonly used in brain and breast imaging

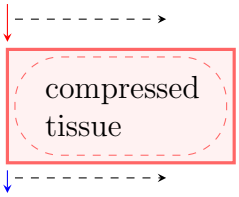

Figure 2.10: Projection shadow: the source and detector move in parallel

Besides geometry considerations there are three different measurement modes: time-domain, frequency domain or continuous wave mode. In time-domain case, the light sources emit pico-second pulses which are broadened by the tissue before reaching the detectors. In case of the frequency mode, the light sources emit an amplitude modulated light signal, which is then scattered by the tissue and detected with smaller modulation depth. In continuous wave mode, the light sources emit time-invariant signals, which are again recorded by the detectors.[38]

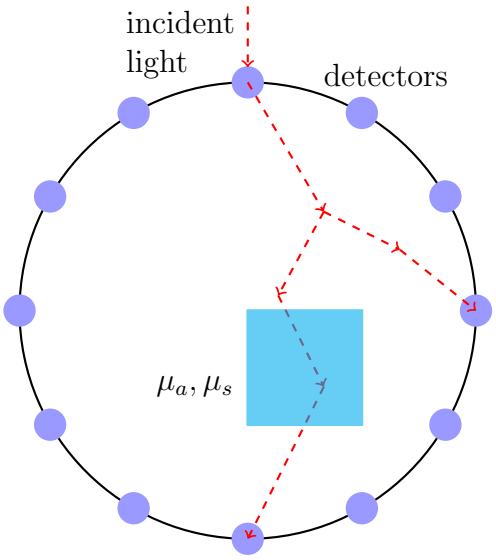

Figure 2.11: Diffuse Optical Tomography (DOT) involves an array of detectors and sources surrounding an object. The incident light is scatter and absorbed by tissues of varying densities. The intensity of photons which are are detected by the surrounding detectors are then used to reconstruct the optical properties of the object being measured.

2.3.2 Theoretical Framework for Diffuse Optical Tomography

Deriving the Diffusion Equation from The Radiative Transfer Equation

The radiative transfer equation (RTE)[39,40,41,42] can mathematically model the transfer of energy as photons move inside a tissue[43,44]. The flow of radiation energy through a small area element in the radiation field can be characterized by radiance $\mathbf{L} = \mathbf{L}(\mathbf{r}, \hat{\mathbf{s}}, \mathbf{t})$. Radiance is defined as energy flow per unit normal area per unit solid angle per unit time. Here, $\mathbf{r}$, denotes position, $\hat{\mathbf{s}}$ denotes unit direction vector and $\mathbf{t}$ denotes time.

$$\frac{\partial \mathbf{L}}{c\partial t} = -\hat{\mathbf{s}} \cdot \nabla \mathbf{L} - \mu_t \mathbf{L} + \mu_s \int_{4\pi} \mathbf{L} P d\Omega + \mathbf{S} \tag{2.25}$$

where c is the speed of light in the tissue, as determined by the relative refractive index[45]. $\mu_t = \mu_s + \mu_a$ is the extinction coefficient, $P(\hat{\mathbf{s}}', \hat{\mathbf{s}})$ is the phase function, representing the probability of light with propagation direction $\hat{\mathbf{s}}'$ being scattered into solid angle $d\Omega$ around $\hat{\mathbf{s}}$. In most cases, the phase function depends only on the angle between the scattered $\hat{\mathbf{s}}'$ and incident $\hat{\mathbf{s}}$ directions. $\mathbf{S} = \mathbf{S}(\mathbf{r}, \hat{\mathbf{s}}, \mathbf{t})$ describes the light source.[46]

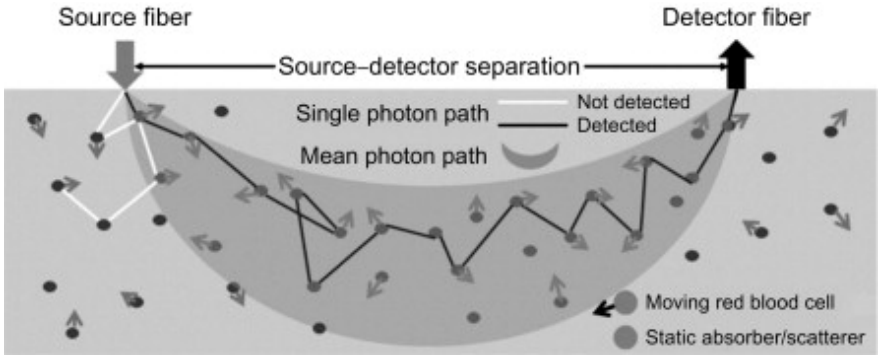

Figure 2.12: Light transport over long distances in tissues as a diffusive process. The photons emitted from the source is scattered or absorbed, with a minority reaching detector fiber. TThe solid line indicates a single photon path in tissue and the shaded "banana" shape represents the mean path of multiple detected photons.[47]

The photons emitted from the source fiber are scattered by static (e.g., organelles, mitochondria) and dynamic (e.g., moving red blood cells) scatterers or are absorbed by absorbers (e.g., hemoglobins, water, lipids), and only some of them are collected by the detector. The

solid line indicates a single photon path in tissue, and the shaded "banana" shape represents the mean path of multiple detected photons. What is known about the pattern of the light path from the light source to the detector is that it follows a banana-shaped curve in which the penetration depth into the tissue is approximately equal to half the distance between the light source and the detector.[38]

Thus by making appropriate assumptions about the behavior of photons in a scattering medium, the number of independent variables can be reduced. These assumptions lead to the diffusion theory (and diffusion equation) for photon transport. Two assumptions permit the application of diffusion theory to the radiative transfer equation[48]:

1. Relative to scattering events, there are very few absorption events. Likewise, after numerous scattering events, few absorption events will occur and the radiance will become nearly isotropic. This assumption is sometimes called directional broadening.

2. In a primarily scattering medium, the time for substantial current density change is much longer than the time to traverse one transport mean free path. Thus, over one transport mean free path, the fractional change in current density is much less than unity. This property is sometimes called temporal broadening.

Both of these assumptions require a high-albedo (predominantly scattering) medium. Given this assumption we can then expand the radiance term making use of it's spherical harmonic basis set, such that

$$\mathbf{L} \approx \sum_{n=0} \sum_{m=-n} L_{n,m}(\mathbf{r}, t) Y_{n,m}(\hat{\mathbf{s}}), \tag{2.26}$$

where $L_{n,m}$ are the expansion coefficients. The term for $n = 0$ and $m = 0$ on the right-hand side represents the isotropic component, whereas the terms for $n = 1$ and $m = 0, 1$ represent the anisotropic component.

Using properties of spherical harmonics and the definitions of intensity $\mathbf{\Phi}$

$$\mathbf{\Phi}(\mathbf{r},t) = \int_{4\pi} \mathbf{L}\, d\Omega \tag{2.27}$$

and current density J,

$$\mathbf{J}(\mathbf{r},t) = \int_{4\pi} \hat{\mathbf{s}}\, \mathbf{L}\, d\Omega \tag{2.28}$$

the isotropic and anisotropic terms can respectively be expressed as follows

$$L_{0,0}Y_{0,0} = \frac{\mathbf{\Phi}}{4\pi} \quad \text{and} \quad \sum_{m=-1}^{1} L_{1.m}Y_{1,m} = \frac{3}{4\pi}\mathbf{J}\cdot\hat{\mathbf{s}} \tag{2.29}$$

Thus we can approximate radiance as

$$\mathbf{L} = \frac{1}{4\pi}\mathbf{\Phi} + \frac{3}{4\pi}\mathbf{J}\cdot\hat{\mathbf{s}} \tag{2.30}$$

Substituting this into the radiative transfer equation and integrating overall the full 4π solid angle, obtains the following scalar differential equation

$$\frac{1}{c}\frac{\partial\mathbf{\Phi}}{\partial t} + \mu_a\mathbf{\Phi} + \nabla\cdot\mathbf{J} = \mathbf{S}. \tag{2.31}$$

While Substituting the diffusion expansion of L into the radiative transfer equation and multiplying both sides by $\hat{\mathbf{s}}\mathbf{s}$, and integrating over the full 4π solid angle, we obtain the following vector differential equation

$$\frac{1}{c}\frac{\partial\mathbf{J}}{\partial t} + (\mu_a + \mu_s')\mathbf{J} + \frac{1}{3}\nabla\mathbf{\Phi} = 0, \tag{2.32}$$

where the transport (or reduced) scattering coefficient μ_s' is given by

$$\mu_s' = \mu_s(1-g) \tag{2.33}$$

and the transport (or reduce) interaction coefficient μ_t' is given by

$$\mu_t' = \mu_a + \mu_s', \tag{2.34}$$

the reciprocal of which is the transport mean free path l_t'.

To derive the diffusion equation it is necessary to further assume that the fractional change in $\mathbf{J}$ within l_t' is small, such that we can argue

$$\frac{\partial \mathbf{J}}{\partial t} = 0. \tag{2.35}$$

With this assumption 2.32 reduces down to Flick's law[49]

$$\mathbf{J} = -\kappa \nabla \mathbf{\Phi}, \tag{2.36}$$

where κ is the known as the diffusion coefficient and $\kappa = \frac{1}{3}(\mu_a + \mu_s')$. By substituting Flick's law into radiative transfer equation we arrive at diffusion equation, given by

$$-\nabla \cdot \kappa(\mathbf{r}) \nabla \mathbf{\Phi}(\mathbf{r}, t) + \mu_a \mathbf{\Phi}(\mathbf{r}, t) + \frac{1}{c} \frac{\partial \mathbf{\Phi}(\mathbf{r}, t)}{\partial t} = \mathbf{S}(\mathbf{r}, t). \tag{2.37}$$

It is worth noting we can always make the substitution $\partial/\partial t \Leftrightarrow i\omega$ to transform between the time and frequency domain.

Introducing Boundary Conditions

As in diffuse optical tomography it is only possible to take measurements on the surface of an object we must introduce a measurement position vector $\mathbf{m}$ which is constrained to a surface $\partial\Omega$ in the domain Ω. As shown in figure 2.11, detectors are placed in an an array on the surface of the object being measured, we can think of $\mathbf{m}$ corresponding to a given detector that lies on the surface of our domain. The outer normal to the surface $\partial\Omega$ at $\mathbf{m}$ is given by $\hat{\nu}$. Realising all of this we can replace $\mathbf{r}$ with $\mathbf{m}$, as positions can only correspond with $\mathbf{m}$. Thus we can represent the radiance captured by a detector as $\mathbf{L} = \mathbf{L}(\mathbf{m}, \hat{\mathbf{s}}, \mathbf{t})$ and

thus intensity at a detector as $\mathbf{\Phi} = \mathbf{\Phi}(\mathbf{m})$.[50]

The boundary condition in radiative transfer equation specify that no photons travel in an inward direction at the boundary, except for source terms, as such

$$\mathbf{L}(\mathbf{m}, \hat{\mathbf{s}}, t) = 0 \quad \text{for} \quad \hat{\mathbf{s}} \cdot \hat{\nu} < 0 \tag{2.38}$$

This condition is difficult to satisfy thus we modify our condition such that we assume that the total inward directed current is zero, which when simplified turns out to be a Robin boundary condition.

$$\int_{\hat{\mathbf{s}} \cdot \hat{\nu} < 0} \hat{\mathbf{s}} \mathbf{L} \, d\hat{\mathbf{s}} = 0 \tag{2.39}$$

$$\mathbf{\Phi} + 2\kappa \frac{\partial \mathbf{\Phi}}{\partial \nu} = 0 \tag{2.40}$$

To incorporate diffuse boundary reflection arising from a refractive index mismatch between and the surrounding medium we include a directionally varying refraction parameter $\mathbf{R}(\hat{\mathbf{s}})$, thus further modify the boundary condition such that the above equations become

$$\int_{\hat{\mathbf{s}} \cdot \hat{\nu} < 0} \hat{\mathbf{s}} \mathbf{L} \, d\hat{\mathbf{s}} = \int_{\hat{\mathbf{s}} \cdot \hat{\nu} > 0} \hat{\mathbf{s}} \mathbf{R} \mathbf{L} \, d^2\hat{\mathbf{s}} \tag{2.41}$$

$$\mathbf{\Phi} + 2\kappa \frac{\partial \mathbf{\Phi}}{\partial \nu} = R \left[\mathbf{\Phi} - 2\kappa \frac{\partial \mathbf{\Phi}}{\partial \nu} \right] \tag{2.42}$$

This can ultimately be rearranged into the Robin boundary condition which ultimately defines the behaviour of the diffuse optical tomography problem at the boundaries.[51]

$$\mathbf{\Phi}(\mathbf{m}) + 2\zeta\kappa(\mathbf{m}) \frac{\partial \mathbf{\Phi}(\mathbf{m})}{\partial \nu} = 0, \quad \text{where} \quad \zeta = \frac{1+R}{1-R}. \tag{2.43}$$

Forward Problem

The forward diffuse optical tomography problem can be written as

$$-\nabla \cdot \kappa(\mathbf{r})\nabla\Phi(\mathbf{r},\omega) + \left[\mu_a \frac{i\omega}{c}\right]\Phi(\mathbf{r},\omega) = 0, \quad \mathbf{r} \in \Omega \tag{2.44}$$

with boundary conditions

$$\Phi(\mathbf{m},\omega) + 2\zeta(c)\kappa(\mathbf{m})\frac{\partial\Phi(\mathbf{m})}{\partial\nu} = q(\mathbf{m},\omega), \quad \mathbf{m} \in \partial\Omega \tag{2.45}$$

where q is a source distribution on the boundary $\partial\Omega$ of domain Ω, modulated at frequency ω. The absorption coefficient μ_a and scattering coefficient μ_a. Th ζ represents the refractive index mismatch, c the speed of light with $\kappa = [3(\mu_a + \mu_s)^{-1}]$.[1]

$$J_n(\mathbf{m},\omega) = -c\kappa(\mathbf{m})\frac{\partial\Phi(\mathbf{m},\omega)}{\partial\nu} \tag{2.46}$$

Ultimately the acquisition process can be thought of as

$$g = \mathbf{M}\Phi, \tag{2.47}$$

where $\mathbf{M}$ here is our measurement operator — the associated operator that acquires measurements from, Φ, the photon density distribution. From 2.44 we understand that the forward model of diffuse optical tomography is governed by a partial different equation, thus we can use the differential operator $\mathcal{L}$ to represent this partial different equation and rewrite this expression as,

$$\mathcal{L}(\mu_a,\mu_s)\Phi = q. \tag{2.48}$$

We can think of this as taking some measurements about some Green's space $\mathbf{G}$, which is

the solution of 2.48.

$$\Phi = \mathbf{G}(\mu_a, \mu_s)\ q. \tag{2.49}$$

Thus we can this of the forward model as being

$$g = \mathbf{M}\ \mathbf{G}(\mu_a, \mu_s)\ q. \tag{2.50}$$

The Inverse Diffuse Optical Tomography Problem

It is worth re-establishing that there is no direct relationship between the absorption coefficient distribution μ_a and scattering coefficient distribution μ_s, (which we can also call our parameters), and the measurements. The problem is nonlinear and multiple or various combinations of (μ_a, μ_s) values can result in identical measurements and are thus indistinguishable .

If we consider this as a mapping between spaces, we are attempting to reconstruct the absorption $\mu_a \in X^{\mu_a}$ and scattering $\mu_s \in X^{\mu_s}$ parameters from a set of measurements $g \in Y$. Though the exact nature of the measurements depend upon the measurement operator M acting on the solutions G acquired via the Green's function. The nature of the measurement operator depend on the discretisation of the space, thus for any set of parameters there are multiple possible of corresponding measurement spaces $Y_1, Y_2, Y_3, ..., Y_N$ which are dependent on their corresponding measurement operator $\mathbf{M}_1, \mathbf{M}_2, \mathbf{M}_3, ..., \mathbf{M}_N$.

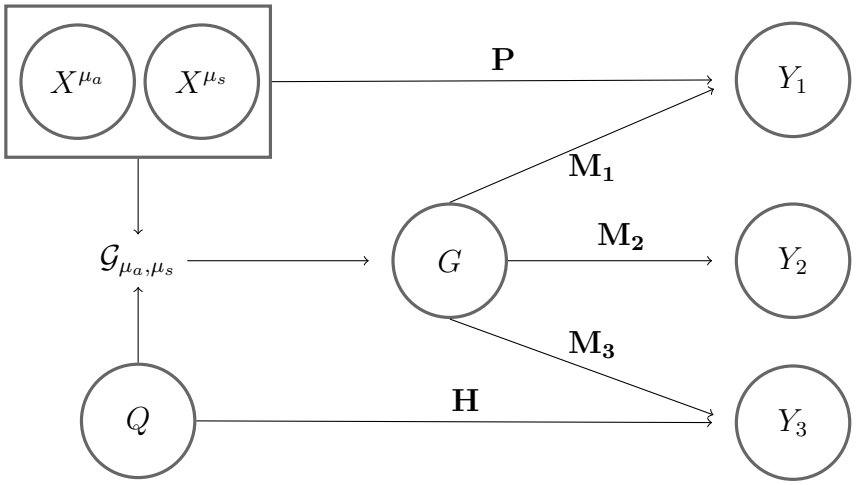

Figure 2.13: The above diagram illustrates visually the operators and associated Banach spaces within diffuse optical tomography[50]

Ultimately the diffuse optical tomography inverse problem can be thought of as reconstructing distribution of model parameters $f = \{\mu_a, \mu_s\}$ from boundary measurements $g = \langle M, J \rangle_{\partial\Omega}$ and is defined by a regularised least-squares approach, where our objective function is

$$\Psi(f) = \underset{f \in \mathbb{R}^n}{\arg\min} \frac{1}{2} \sum_i [g_i - A_i(f)]^2 + \tau \mathcal{R}(f), \tag{2.51}$$

where the forward model A is defined by the earlier set of partial different equations. $\mathcal{R}$ and τ are regularization and hyper-parameters respectively, which can be chosen to take advantage of prior knowledge. In this scenario we can make use of reconstruction techniques such as the Gauss-Newton algorithm.[52]

Chapter 3

Method and Experimental Design

In this thesis we use the reconstruction from the Gauss-Newton algorithm as our gold standard and compare the performance of multiple deep learning architectures.

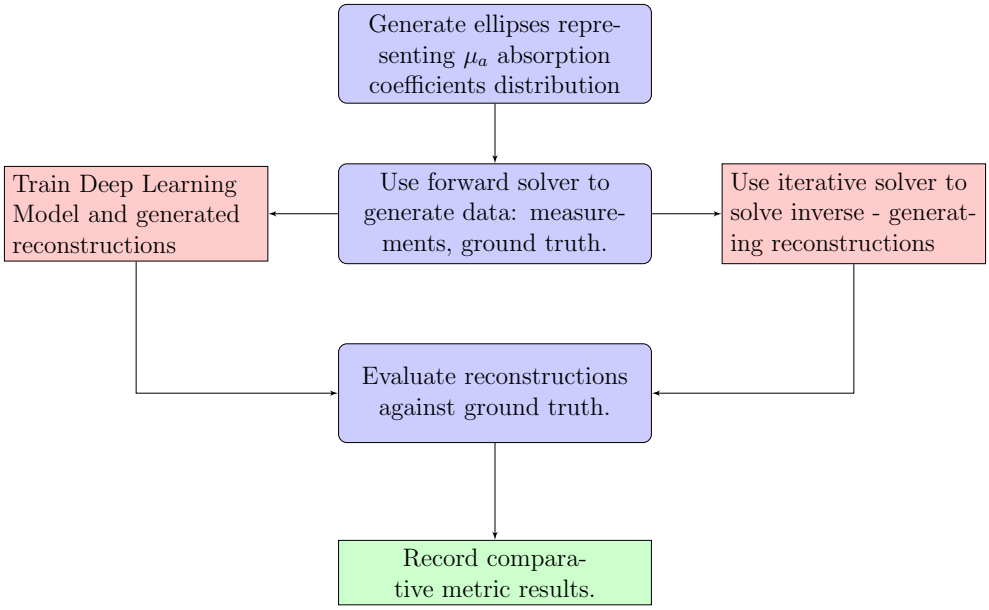

Figure 3.1: Flow diagram showing experimental design

Figure 3.1 provides an overview of the experimental method followed. In summary we first generate a number of images composed of multiple overlapping ellipses, which represent the μ_a absorption coefficients distributions; (the set of images are checked for duplicates which are deleted. The images are fed into a forward solver which then generates our data-set;

images and the ground truth. (If noise is required, it is added to our set of measurements). A reconstruction method is then chosen and a set of reconstructions are generated and compared against their corresponding ground truths.

In the case of deep learning methods, the data-set is split 8:2 into a training and test set, the network is then trained on the training data, reconstructions are then generated using the test set and compared against their ground truths.

All experiments were run on a ROG Zephyrus G14 Laptop, (CPU: AMD Ryzen 7 5800HS with Radeon Graphics 3.20 GHz, GPU: NVIDIA GeForce RTX 3050 Ti, RAM: 16.0 GB RAM, System Type: 64-bit operating system, x64-based processor), running a Windows 11 Home operating system. The Python environment had the following dependencies: NumPy 1.21.5, Scikit-image 0.19.2, Scikit-learn 1.1.1, PyTorch 1.11.0. Additional software used include MATLAB R2022a and Toast++ v2.0.2.

3.1 Data Generation

Generating the synthetic data involves 2 steps; generating thousands of images with overlapping ellipses and feeding those images into a forward solver. The images of overlapping ellipses are generated using NumPy in Python. These images were fed into Toast++, which solved the forward problem using a finite element method (FEM).

The basic concept of finite element can be thought of as splitting the domain Ω into smaller individual patches, which we call finite elements, and finding local solutions that satisfy the differential equation within the boundary of this patch. By stitching together these individual solutions a global solution can be obtained.[53] This allow us to capture local effects while also representing the total solution. Thanks to this method we can also include the ability to include objects with dissimilar material properties within the domain.[54]

A forward solver was built from scratch using MATLAB's partial differential equations toolbox, though it was eventually found that the toolbox did not allow for complex boundary

conditions — such as the Robin, Dirichlet-to-Neumann or Neumann-to-Dirchlet boundary conditions. A data-set for electrical impedance tomography was also attempted using the solver EIDRORS, though EIDORS did not allow for images to be imported and mapped to a mesh.

Data-sets of multiple resolutions and sizes were initially created, we found that 16×16 pixels provides distinguishable ellipses while also minimizing memory consumption. It was found that the MNIST data-set was composed of thin lines that could not be reconstructed. Reconstruction requires objects with a minimum diameter and performs best on robust shapes such as circles, squares and ellipses. Lines are fragile, as a result their reconstruction would end-up appearing as points; as was the case with the MNIST data-set.

The use of elliptical phantoms has been shown to be effective in the past when developing data-driven methods to solve inverse problems.[55] Thus elliptical phantoms were used for both training and the test data-sets. Unfortunately it was found the elliptical phantoms did not provide enough generality for the models to reconstruct the Shepp-Logan phantom, so validation was performed only on the test set – which shared similar properties to the training set.

To generate the training and test data-sets we firstly start by creating a 16×16 pixel canvas and randomly determine the number of ellipses, between 1 and 5, to place onto the canvas. To ensure no ellipses is clipped by the size of the canvas a margin is created which is 10% the dimensions of the canvas.

Once an ellipse is randomly placed within the canvas it is necessary to determine its position and then determine the maximum height and width the ellipses can have without being clipped by the boundaries of the canvas. Both the position and dimensions of each ellipses are randomly taken from a uniform distribution.

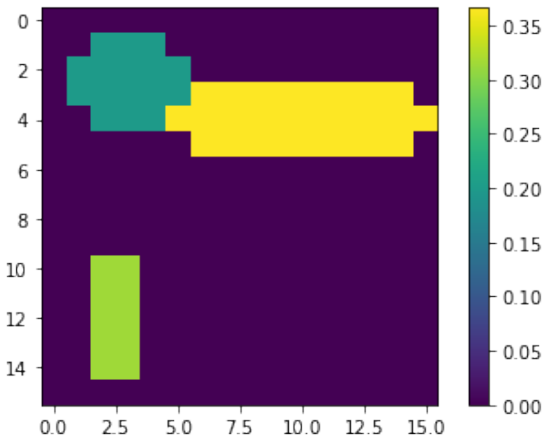

Figure 3.2: Ellipses generated in python, representing our absorption coefficients distribution μ_a which are attempting to reconstruct

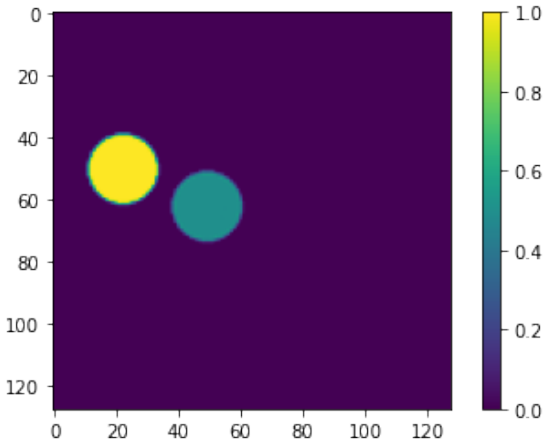

Figure 3.3: Unchanging image with circles which represent our scattering coefficients distribution μ_s which remains constant. This image is resized to be of the same dimensions as μ_a.

These generated images of ellipses represent the absorption coefficients distribution μ_a, which are the solution parameters that we are attempting to recover. A mask is applied to μ_a and μ_s, clipping the image to fit the domain Ω which we take measurements about. Initially the domain Ω is discretised by a 16×16; identical to the dimensions of μ_a images.

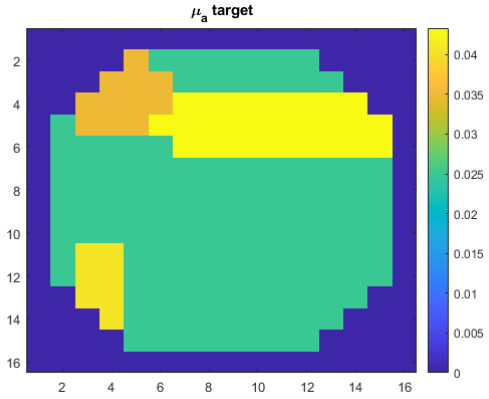

Figure 3.4: μ_a is discretised and a solution mask is applied so that it can be mapped to the mesh covering the circular domain, where the detectors and sources are placed at the boundary.

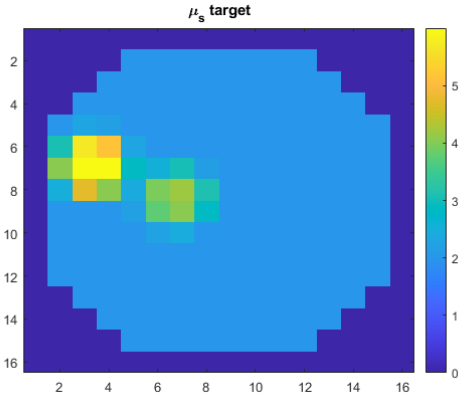

Figure 3.5: In our experiments μ_s remains unchanged

After both μ_a and μ_s are fully discretised by a regular grid and had solution mask applied to them — clipping them so that so that it can be fully mapped to a circular domain, they are both then mapped onto a fine mesh, where sources and detectors surrounded the perimeter

of the mesh.

The initial circular mesh has a radius of 25mm. The circle is discretised by 6 radial sectors and 32 equally spaced rings. Notice the domain Ω is tessellated into triangles with finer discretisation at the boundaries. The boundary layer is composed of 2 further rings and more finely discretised improving the accuracy of measurements acquired by detectors at the boundary $\partial\Omega$ — sources and detectors are denoted by red and blue circles on boundary, respectively. To generate our data-set we used 8 sources and 8 detectors.

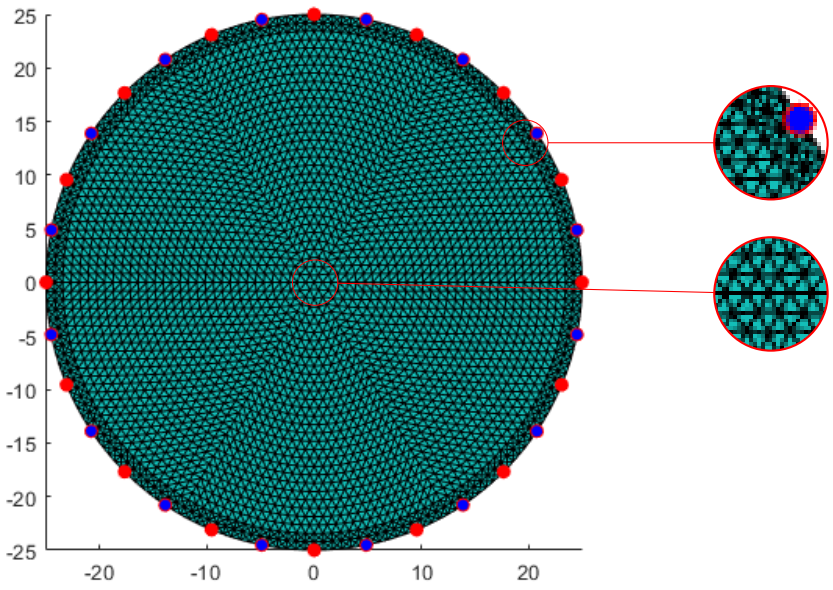

Figure 3.6: Discretisation of our mesh; 25mm radius with 6 radial sectors, 32 rings and 2 boundary rings. Notice the domain Ω is tessellated into triangles with finer discretisation at the boundaries improve accuracy of boundary measurements. Sources and detectors are denoted by red and blue circles on boundary, respectively.

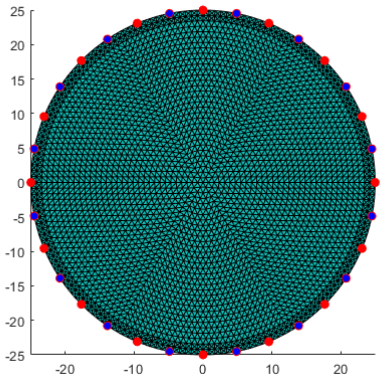

Figure 3.7: Mesh which discretizes our parameters, the boundary is surrounded by an array of sources and detectors

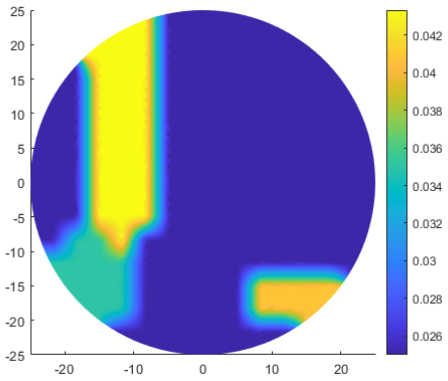

Figure 3.8: μ_a is mapped to the mesh

Once the μ_a absorption and μ_s scattering coefficients distribution are fully mapped to the mesh, we can then solve the forward problem by using a finite element method, this in turn generates the measurement data that we are interested in. Both μ_a and μ_s are mapped to mashes with identical discretisations, though we are only interested in reconstructing μ_a.

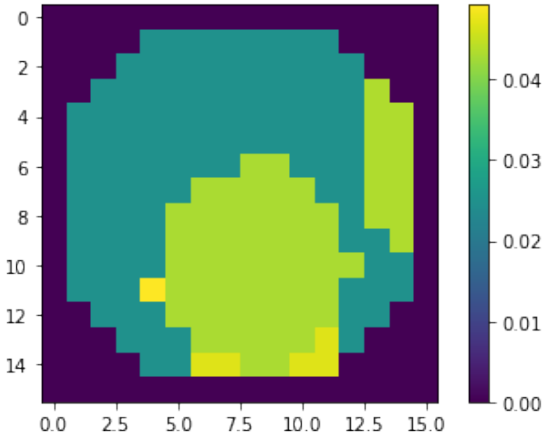

Figure 3.9: 16×16 matrix which represents our μ_a parameter

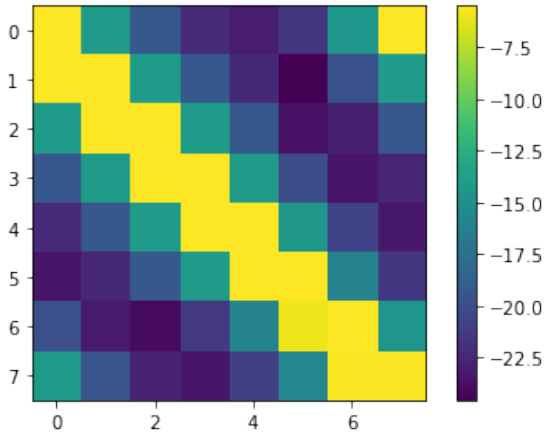

Figure 3.10: 8×8 matrix representing intensity measurements from 8 detectors

Figure 3.10 represents a single set of measurements associated with a single set of μ_a absorption coefficients, as shown in . To be able to interpret these set of measurement it is necessary to recall that we have a total of 8 sources and 8 detectors at the boundary of our domain. Each source projects a beam into the domain, the reflections of which are picked up by all 8 detectors. This processed is repeated for all 8 sources. Each column corresponds

to a different source and each row corresponds to a reading from one of 8 detectors.

In general the measurements acquired from equation 2.44 are complex, where $\Phi(\mathbf{m})$ represents the complex-valued photon density distribution. We can ensure that our output is purely real by setting our modulation frequency to zero. (Or we could only use the real component of the measurements as they are orders of magnitude greater than the imaginary component).

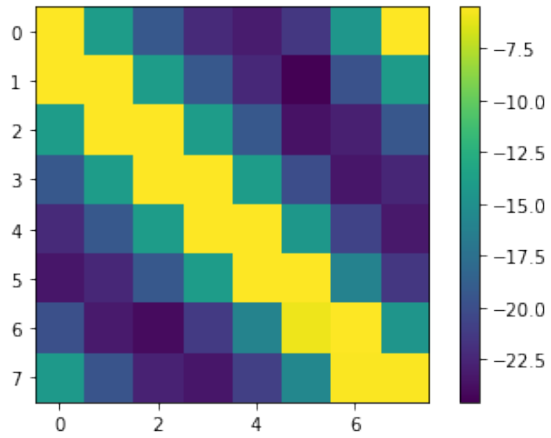

Figure 3.11: 8×8 matrix which represents the real component Φ_{Re} of the complex-valued photon density.

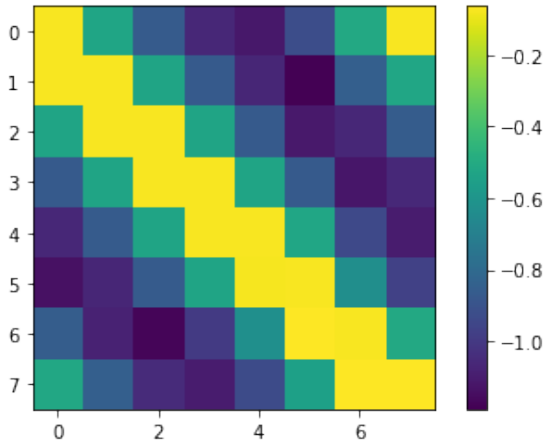

Figure 3.12: 8×8 matrix which represents the imaginary component Φ_{Im} of the complex-valued photon density.

$\Phi(\mathbf{m})$ our complex-valued photon distribution can be separated into $\Phi = \Phi_{\text{Re}} + \Phi_{\text{Im}}$, thus for completeness we have shown that the real component of the complex-valued photon density distribution is orders of magnitude greater than the imaginary component as seen in figures 3.11 and 3.12 an in effect be ignored either when taking the absolute values or taking purely the real values.

Within our experiments we used a refractive index mismatch of 0.4 and modulation frequency of zero. As well as using a background μ_a coefficients distribution of 0.025 and μ_s scattering coefficients distribution of 2.

3.2 Comparison Metrics

To compare the quality of reconstruction between techniques comparison metrics are necessary. We make use of 3 metrics to determine the quality of the image reconstruction; root-mean-square error (RMSE), structural similarity index (SSIM) metric and peak signal-to-noise ratio (PSNR).

RMSE provides a accurracy metric which is always non-negative. A value of zero, which in practice is never achieved, would indicate a perfect fit to the data. Though generally a reconstruction with a lower RMSE value suggests it is closer to the ground truth than a reconstruction with a higher value; assuming we are comparing the ground truth to a reconstruction. The below compares two images of size $m \times n$; the ground truth $f(i,j)$ to its reconstructions $f^{\#}(i,j)$.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=0}^{m-1} \sum_{\mathbf{j=0}}^{n-1} (f(i,j) - f^{\#}(i,j))^2}{n \times m}}. \tag{3.1}$$

A major disadvantage of using RMSE as a comparison metric, is that it is sensitive to outlier, as the effect of each error is proportional to the size of the squared error, causing larger errors have a disproportionately large effect.

PSNR on the other hand is the ratio between the maximum possible pixels value MAX_I of an image and the corrupting noise within that image. Due tot the wide dynamic range within an image, PSNR is usually expressed as a logarithmic quantity using the decibel scale.

$$\text{PSNR} = 20 \log_{10} \left(\frac{\text{MAX}_I}{\text{RMSE}} \right), \tag{3.2}$$

SSIM provides attempts to measure the similarity between two images by extracting and comparing 3 key features from an image; luminance, contrast and structure. SSIM provides us with a value between -1 (no similarity) and 1 (fully similar). The metric is calculated on various windows of an image. Here x corresponds to a window on our ground truth image

and y is a window on our reconstructed image. We are comparing the similarity between a shared $N \times N$ window across both images.

$$\text{SSIM} = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}. \tag{3.3}$$

The average pixel intensity of the corresponding windows is denoted by μ, while the standard deviation is denoted by σ and the covariance of the two image is σ_{xy}. In essence c_1 and c_2 act as stabilisers.

3.3 Reconstruction Algorithms

3.3.1 Gauss Newton

The Gauss Newton method, as outlined in algorithm 2, acts as our gold standard for reconstruction. The iterative reconstruction on average takes 29.12 seconds per image. (This was measured over 100 images). The maximum number of iterations the Gauss-Newton loop performs is set to 100, with the loop ending if the RMSE error between the measurement data and the measurement data acquired from the reconstruction is below 1×10^{-4}.

The Gauss Newton algorithm was implemented using Toast++ and MATLAB. MATLAB's own generalised minimal residual method (GMRES) solver was used with a basic line search to find the step size τ_k. After each each step the new reconstruction is passed into the forward operator generating a set of measurements. We can then compare input measurements with the generated measurements as shown in equation 2.51 to determine how accurate the reconstructions of μ_a are. We made use of the Total Variation (TV) regularization within Toast++, which enforced a smoothness condition on the solution.

3.3.2 Deep Learning

Between deep learning reconstruction methods hyper-parameters remained constant. To compare performance between deep learning methods it was necessary to set the the number of epochs to 20. This in essence acted as a stopping criteria and allowed experiments to be performed on a local machine in a timely manner. The Adam optimiser, which takes in multiple hyper-parameters, was used across deep learning methods. These were set to step size = 50, learning rate = 0.001, gamma = 0.5 and weight decay = 1e-4. A standard batch size of 10 was also used to train a given network.

Between deep learning implementations we maintain the same data-set of 10,000 data-points; the data was split 8:2 between a training set and test set. Each measurement data was 8×8 with the corresponding ground truth image being of size 16×16. This avoided any bias that could be introduced by varying data-sets. The smaller data-set size allowed for faster training and evaluation. By comparing identical image reconstruction, when it was possible, it gave a visual understanding of how each network was improving image reconstruction.

Multilayered Perception

In the case of the multilayered perceptron both the measurement data-set and the ground truth data-set was first vectorised before being fed into the network. Thus our input layer was 64 nodes while our output layer was 256 nodes. This ultimately provided us with a baseline of what deep learning architectures were capable of. The number of hidden layers were incrementally increased and various network widths were tested.

Convolutions Neural Network

The convolutional neural network architecture provided us with some difficulties, as the the input measurements were 8×8 while our ground truth was a 16×16 square matrix. To overcome this the input data was fed through a fully-connected layer which projected the vectorised sets of measurements from 8×8 to a 16×16 representation — the representation

was shaped to have a single channel.

This was then passed through a convolutional layer that extracted low level details. This convolutional layer had a single input channels and 128 output channels as well as a kernel size of 5, stride of 1 and padding of 2.

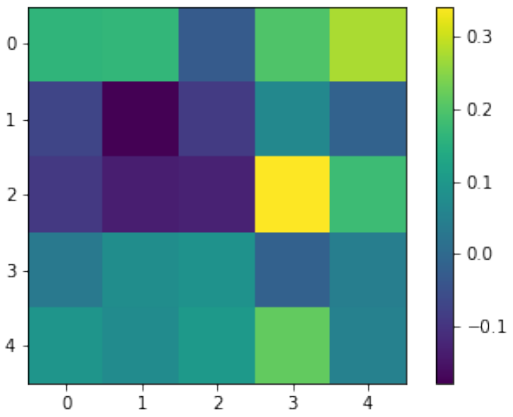

Figure 3.13: 5x5 kernel

The output from this convolutions layer was then fed through a series of smaller convolutional layers with kernels with a size of 1, stride of 1 and zero padding. The final convolutional layer had an input channel of 128 and output channel of 1, thus returning the image to its original number of channels. Other than the last convolutional layer, each convolutional layer with a kernel of 1 had 128 input channels and 128 output channels, thus channels of the data were projected form 1, to 128 and back to 1.

Fourier Neural Operator Implementation

Our measurement data-set consisting of 8×8 matrices is fed into the network. The measurements then passes through a fully connected layer which significantly increases the number of trainable parameters by acting on each individual pixel within the input image; this effecitvely increases the number of channels of our data-set. In our case we decided upon the width of our network to be 64 and thus data then passes through a fully connected

layer increasing the number of training parameters to 2 $\times$ width. If we take a moment to consider what the first layer is in fact achieving, the first layer of the network acts as an operation which generates $v^{[0]}$, such that $\mathrm{NN}_1(g(x), x) \to v^{[0]}$. This is completed point-wise and thus for every measurement i in the set of measurements N, 64 trainable parameters are generated.

Following this $v^{[0]}$ passes through two independent layers — a spectral layer and a convolution layer (the combinations of which is what is earlier denoted as a Fourier layers). Within in the spectral layer a discrete Fourier Transform for real input is applied to v_0, such that only positive frequencies are outputted (and the negative-frequency terms are rendered redundant). Frequencies above $k_{\max}$ are cut-off, truncating the total signal such that $\mathscr{F}\{v^{[0]}\} \to \widetilde{v}^{[0]}_{ijk}$. Which is then used to update the weights in R, this can be generalised for all $v^{[n]}$ such that

$$R^{[n+1]}_{ilk} = \widetilde{v}^{[n]}_{ijk} R^{[n]}_{jlk}\,, \qquad k = 1, \ldots, k_{\max}. \tag{3.4}$$

Through experimentation it was found that for a given dimension the maximum number of Fourier modes that we can input into the network is $k_{\max} = (\dim/4 + 1)$; where here dim refers to the dimensions of the image. Thus if we are considering purely the width of the input image which is 8, the maximum number of Fourier modes we can achieve is 5. This appears to be a limitation with PyTorch. The Fast Fourier transform generates complex values which a majority of PyTorch's functions are unable to handle. To get around the network makes use of PyTorch implementation of Einstein summation convention which can handle complex values. (Though this comes with its own limitations as PyTorch's implementation Einstein summation convention requires very explicit instructions).

Chapter 4

Results and Discussion

Our results form the Gauss-Newton algorithm acts as our gold standard. This is compared against other deep learning methods. The minimum bar for deep learning methods is set by the multi-layer perception as it is the earliest and least complex example of a deep learning feed-forward algorithm. Overall we found that the Fourier neural operator architecture outperformed the Gauss-Newton algorithm as well as all other tested deep learning architectures. Performance of the Fourier neural operator improved as the number of Fourier layers were increased.

4.1 Results from Gauss-Newton

As mentioned earlier the Gauss-Newton algorithm is an iterative reconstruction algorithm. It has a significantly faster compute time over comparable iterative reconstruction algorithms as it does not require the calculation of second derivatives.[4]

It took 42 minutes and 22.12 seconds to reconstruct 100 images, with the average 16×16 pixel reconstruction taking 25.64 $\pm$3.77 seconds to complete, and requiring 22.62 $\pm$3.28 iterations for it to achieved an RMSE error below 0.0001. Generating 1049 images, which henceforth are used as the gold standard for reconstructions, took over 8 hours.

The reconstruction results were acquired for 1049 images with each image representing a

different absorption coefficients distribution μ_a. Below in table 4.1 we recorded the quality of the iterative reconstruction achieved by the Gauss-Newton algorithm. These sets of results act as our benchmark. To provide context the means and associated standard deviation were recorded for each one of the following metrics: peak-signal-to-noise ratio (PSNR), root-mean-squared error (RMSE) and structural similarity index metric (SSIM). These three metrics are widely used standard image quality metrics.[56]

RMSE	**PSNR**	**SSIM**
0.00233 $\pm$0.00119	27.17938 $\pm$3.25252	0.91126 $\pm$0.02928

Table 4.1: The table records the quality of the iterative reconstruction achieved by the Gauss-Newton algorithm. The means and associated standard deviation are recoreded for each one of the metrics: peak-signal-to-noise ratio (PSNR), root-mean-squared error (RMSE) and structural similarity index metric (SSIM). The reconstruction results were acquired for 1049 images with each image representing a different absorption coefficient distribution μ_a.

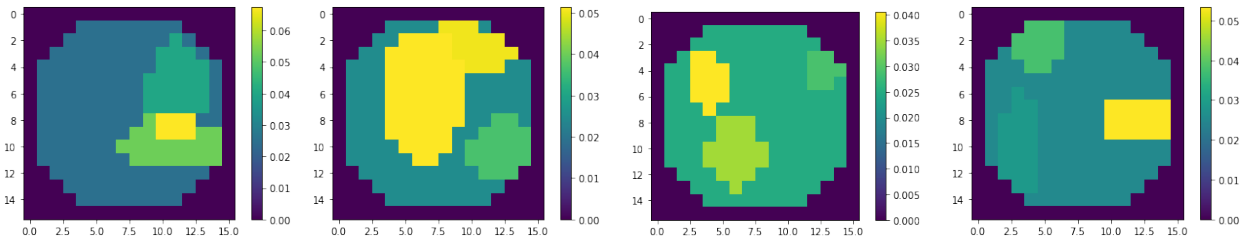

Figure 4.1: Ground truth images for the Gauss-Newton Algorithm

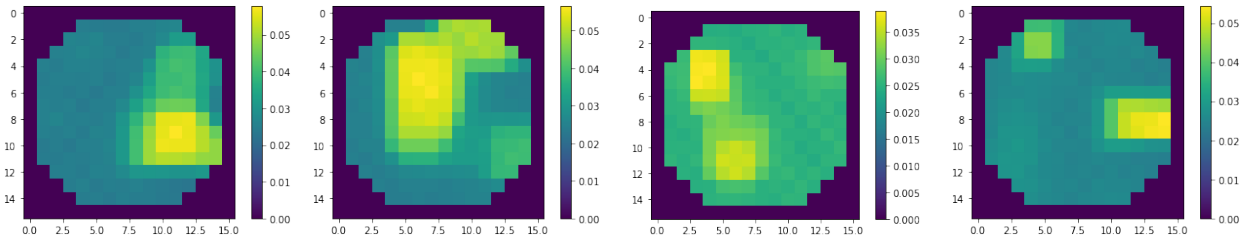

Figure 4.2: Corresponding reconstructions from Gauss-Newton iterative reconstruction

To better understand the performance of the Gauss-Newtons reconstruction, and to appreciate how deep learning techniques compare against it, we should investigate the nature of outliers and thus the absorption coefficients μ_a distributions which the Gauss Newton iterative algorithm struggles with. (A short refresher to box plots and outliers is provided within the appendix).

Statistical parameters such as the mean, standard deviation associated with each of the three image quality metrics, provides the reader with a near complete analysis of the performance of each algorithm in relation to one another. As a result we have plotted the reconstruction performance of the Gauss-Newton algorithm in the box-plots shown in figure 4.3 .

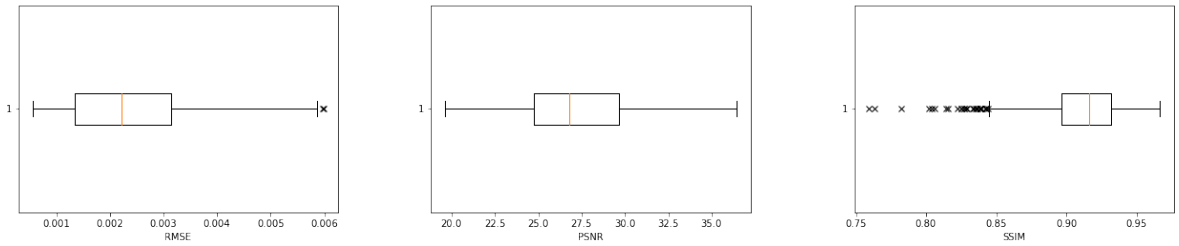

Figure 4.3: Boxplots for the RMSE, PSNR and SSIM of the Gauss-Newton Algorithm — within that order.

Immediately from the box plots in figure 4.3 we can determine that for the Gauss-Newton algorithm not a single image representing an absorption coefficient distribution μ_a, (which henceforth we will merely call μ_a images), was considered a PSNR outlier. Thus we can determine the Gauss-Newton algorithm does not give preferential treatment to certain geometries when it comes to reconstructing the peak signal.

Though in the case of RMSE, out of 1049 μ_a images there were 2 μ_a images where their reconstructions were consider to be RMSE outliers. In this case an RMSE outlier was any reconstruction with an RMSE values above 0.00586.

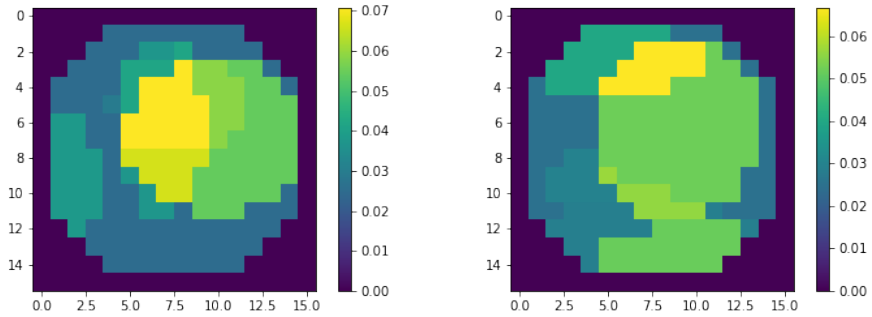

Figure 4.4: The ground truths for images where their reconstructions had RMSE values that were considered outliers.

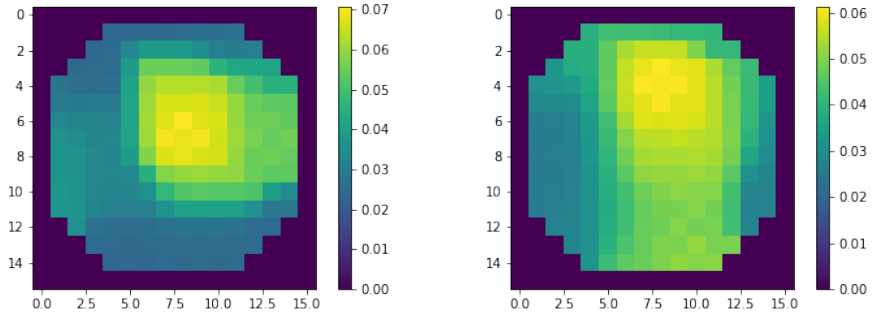

Figure 4.5: The corresponding reconstructions for images where their reconstructions had RMSE values that were considered outliers. The corresponding RMSEs are (left) 0.00597 and (right) 0.00598

In the case of SSIM the gauss-newton algorithm clearly found certain μ_a images to be preferential, as there were a total of 35 SSIM outliers, with the worst outlier having a SSIM value of 0.75930. (In the case of SSIM at outlier was any μ_a image with a reconstruction that had SSIM value lower than 0.84422).

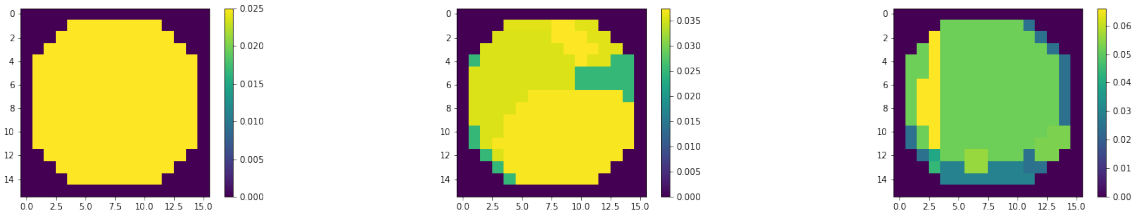

Figure 4.6: Ground truth for Gauss-Newton SSIM outliers.

Based on the μ_a images in figure 4.6 and the reconstructions in figure 4.7 it is clear that there are certain μ_a coefficients distributions which the algorithm struggles with. Distributions which are mostly composed large ellipses which consume a majority of the μ_a image. There also appears to be some checkered artefacts within the reconstruction, which appears to be due to the Gauss-Newton algorithm being run on the mesh, where the reconstruction is mapped back onto the grid — as these artefacts do not appears of the reconstruction on the mesh.

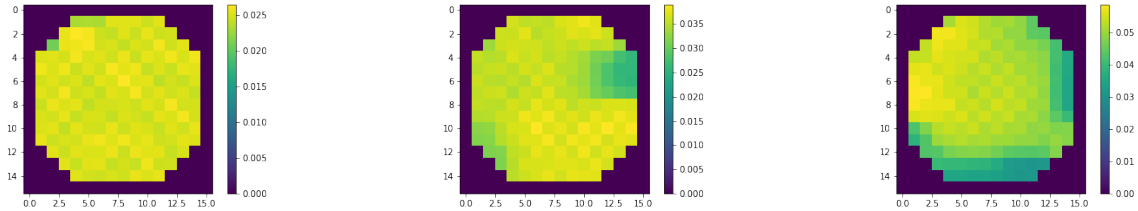

Figure 4.7: Corresponding reconstruction and their SSIM values: 0.83889 (left), 0.82483 (centre) and 0.0.75930 (right).

4.2 Results for Multilayered Perceptron

Within the field of deep learning, we can think of multilayer perceptron (fully-connected network) as providing us with a baseline or minimum performance benchmark for the quality of reconstruction achievable using a standard neural network architecture.

In comparison with the iterative Gauss-Newton reconstruction method used earlier, the inference time capable of deep learning methods are by magnitudes faster. In the case of the multilayer perceptron, it achieved an inference time of 0.81991 seconds. Which means that, after having been trained it was able to reconstruct 2000 μ_a images in 0.81991 seconds. (This was achieved for an a model with 3 layers and a width of 300 nodes).

Based on universal approximation theorem for operators we know it is possible to learn the operator with a single hidden layer, thus it naturally occurs to investigated the added value extra layers provide. These experiments were recorded in table 4.2. We increased the number of layers and width at each layer until training failed, thus determining the maximum ratio between number of parameters and training data size at a given network depth.

No of Hidden Layers	Width	No of Parameters	Training Time/sec	RMSE	PSNR	SSIM
none*	none*	16640	22.03	0.00884 ±0.00196	15.43671 ±1.17854	0.48292 ±0.06402
1	100	32356	29.51	0.0035 ±0.00149	23.36937 ±2.25283	0.80938 ±0.04717
	300	96556	31.6	0.00343 ±0.00135	22.60682 ±2.56179	0.81176 ±0.04663
	400	128656	33.38	0.00330 ±0.0015	23.83111 ±2.74431	0.81915 ±0.03948
2	100	42456	38.7	0.00383 ±0.00161	21.79356 ±2.51705	0.78361 ±0.05199
	300	186856	39.08	0.00378 ±0.00132	22.40158 ±1.75336	0.79218 ±0.05461
	400	289056	41.28	0.00372 ±0.00145	22.07932 ±2.12344	0.80674 ±0.04839
	500	411256	43.72	0.00334 ±0.00147	23.59036 ±2.4161	0.81404 ±0.04311

Table 4.2: The table records the quality of the reconstruction achieved by the multilayered perceptron architectures, with varying node widths and number of layers. The means and associated standard deviation are recorded for each one of the metrics. *The network has no hidden layers and thus not a specified width.

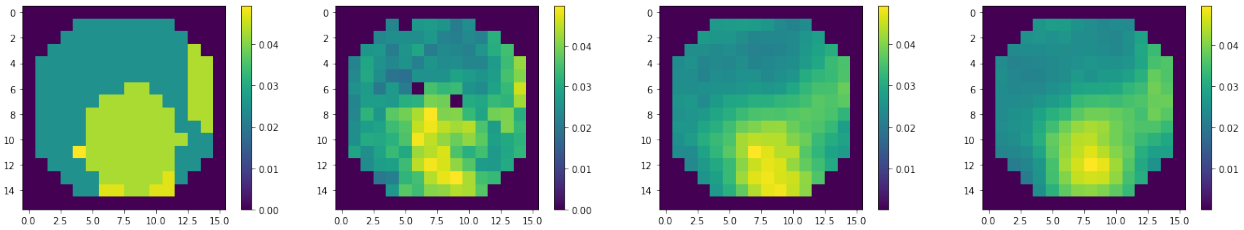

Figure 4.8: Selected reconstruction from multilayer perceptron for varying network architectures: From left to right: Ground truth, 1 layer, 2 layers 100 nodes width, 2 layers 400 node width

To investigate the reconstruction ability of the multilayered perceptron further we focused on an architecture with a the single hidden layer and a width of 400 as it had the lowest RMSE value and best SSIM value. For this architecture there were three outliers which all have RMSE values above 0.00776.

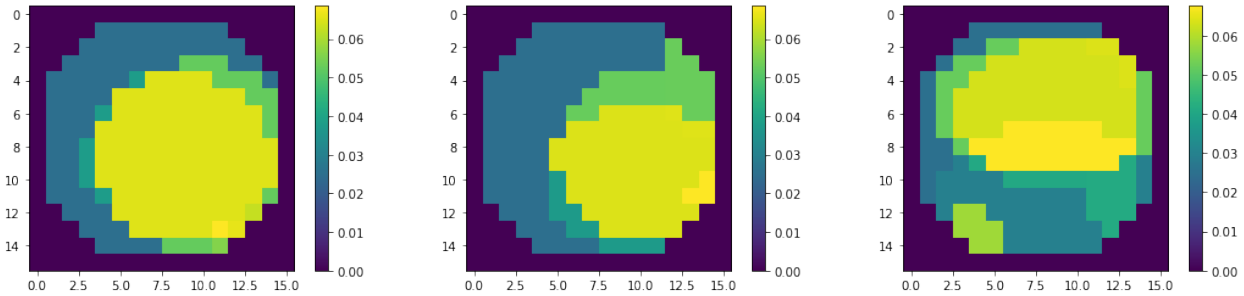

Figure 4.9: Ground truth for multilayered perceptron RMSE outliers.

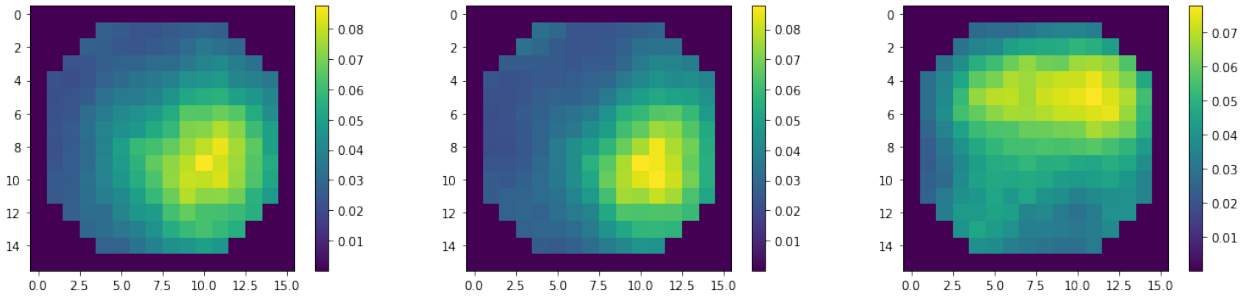

Figure 4.10: Reconstruction RMSE: 0.00950 (left), 0.00851 (centre) and 0.00809 (right).

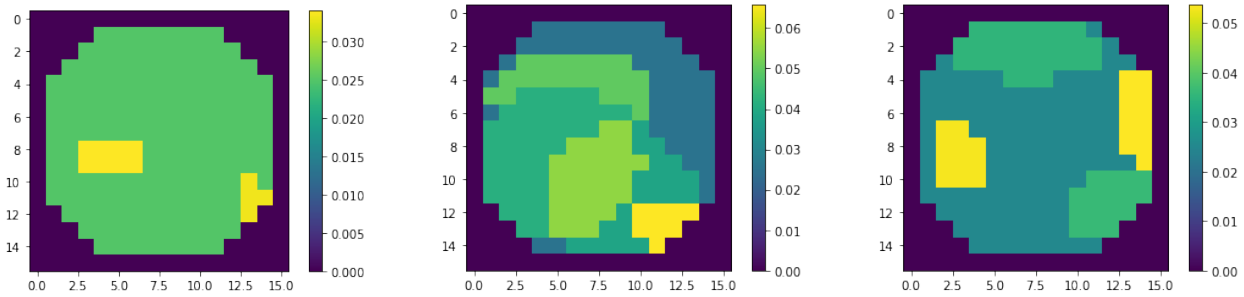

Figure 4.11: Ground truth for multilayered perceptron SSIM outliers.

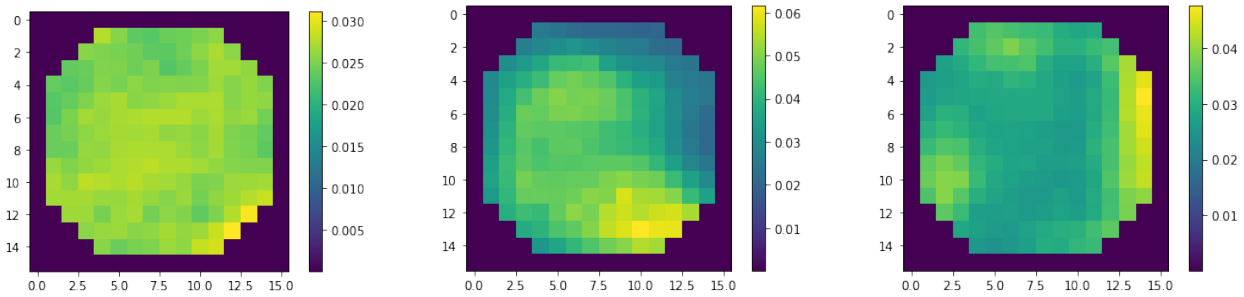

Figure 4.12: Reconstruction SSIM: 0.66750 (left), 0.67894 (centre) and 0.66953 (right).

The multilayered perceptron had 5 μ_a image reconstructions with an SSIM values which were below 0.68299 and considered outliers. Though the network showed no preferential treatment when it came to the reconstructions PSNR values μ_a.

4.3 Results for Convolutional Neural Network

An immediate dramatic advantage of moving the a convolutional neural network architecture is a reduction is trainable parameters — we are training a fifth of the number of parameters

by moving from an 2 layered multilayered perceptron (with a width of 400) to a 2 layered convolutional neural network. Despite the reduce number of parameters, training time almost doubles. Within a convolutional neural networks, parameters are shared learned kernel, though are used multiple times across the image. Every convolution is essentially a multiple nested loop. This means that dense layer needs a fraction of the time to train compared to convolutional layers.

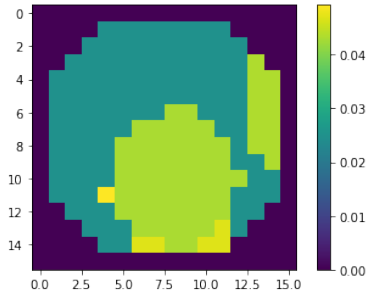

Figure 4.13: Ground truth for CNN

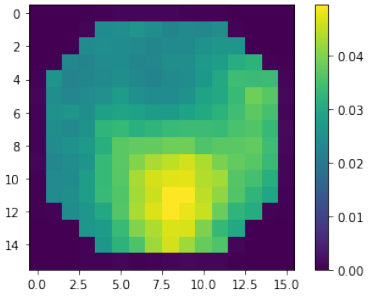

Figure 4.14: Output from 2 layer 2D CNN.

No of Convolution Layers	No of Parameters	Training Time/sec	RMSE	PSNR	SSIM
2	20097	51.01	0.00344 ±0.00150	23.38329 ±2.66768	0.81959 ±0.04338
3	36609	63.18	0.00339 ±0.00151	23.50601 ±2.90928	0.82202 ±0.04463

Table 4.3: The table records the quality of the reconstruction achieved by the convolutional neural network. The means and associated standard deviation are recorded for each one of the metrics: peak-signal-to-noise ratio (PSNR), root-mean-squared error (RMSE) and structural similarity index metric (SSIM). The reconstruction results were tested 2000 images with each image representing a different absorption coefficient distribution μ_a.

One note worthy advantage of using the convolutional neural network is that we immediately we see a noticeable improvement in reconstructions SSIM values despite RMSE and PSNR values deteriorating (when we compare it with the 2 layer multilayer perceptron with a width of 400 nodes). Which makes sense when we consider that SSIM is in fact a perception-based model that considers image degradation as perceived change in structural information, while other metrics merely tell us absolute error.

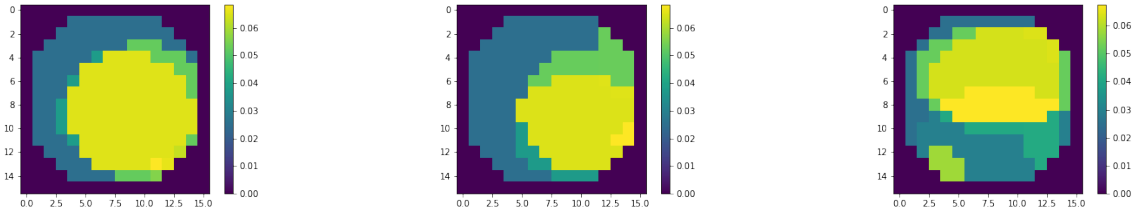

Figure 4.15: Ground truth for CCN RMSE outliers.

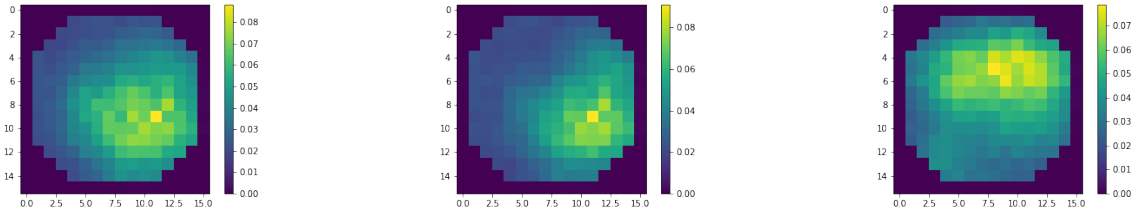

Figure 4.16: Reconstruction RMSE: 0.00982 (left), 0.00898 (centre) and 0.00893 (right).

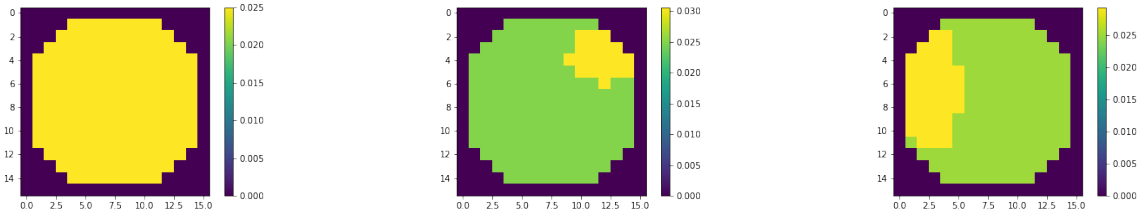

Figure 4.17: Ground truth for CNN PSNR outliers.

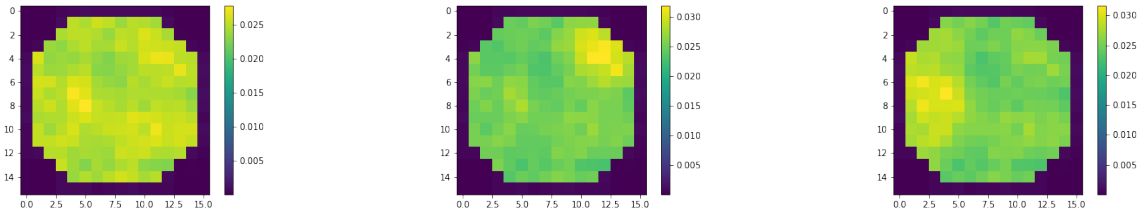

Figure 4.18: Reconstruction PSNR: 30.08536 (left), 30.11678 (centre) and 30.09683 (right).

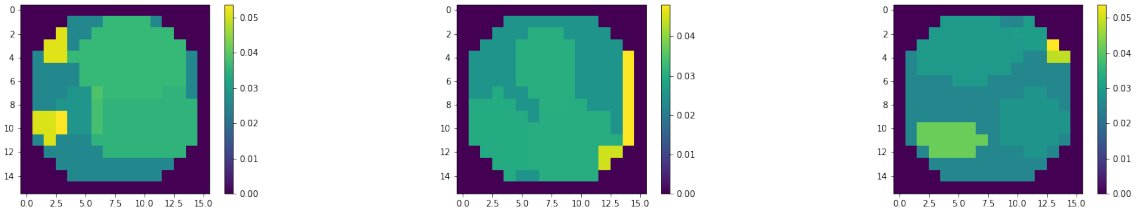

Figure 4.19: Ground truth for CNN SSIM outliers.

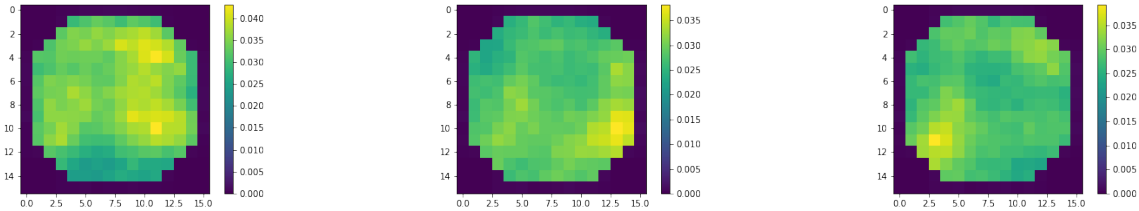

Figure 4.20: Reconstruction SSIM: 0.60506 (left), 0.59968 (centre) and 0.64810 (right).

The network has a preference for μ_a images with few ellipses and simple geometries, reconstructing them with a far stronger PSNR values. Every PSNR outlier, (with a value above 064330 there were 12 of), had far less complexity and fewer ellipses than the rest of the data-set. In fact μ_a images with complex geometries composed of multiple ellipses had a significantly worse SSIM value (11 of which were below 0.66601), in the case when multiple ellipses overlap the RMSE value is significantly worse than the rest of the reconstructions, with 3 being above 0.00779.

Though the problem may be ill-posed the networks has a strong preference for simple geometries, further confirming the convolutional neural networks ability to learn spatial dependencies, calculating features for different pixels based on their neighboring pixel. The filters within a trained convolutional neural network would have learnt local information associated with the measurement data set, thus the networks proportionally high SSIM values and strong metric scores, despite the relatively low number of parameters.

4.4 Results for Fourier Neural Operator Architecture

One would expect that that the quality of reconstruction to improve as the number of Fourier layers increase. To investigate the impact of the number of Fourier layers on the quality of the reconstruction we compared reconstructions from 5 different architectures — one, two, four, six and eight. All models were trained on the same data-set and hyper-parameters remained untouched. The various reconstructions can be seen within the appendix, though a select few are provided below to demonstrated how changing the number of Fourier layers impacts the reconstruction.

The hyper parameters were as follows: batch-size 10, learning rate $\alpha = 0.001$, step-size 50, gamma which is the momentum of the decay term 0.5, number of Fourier modes 5, network width 64 and weight decay 10^{-4}, number of epochs 20.

No of Fourier layers	No of Parameters	Training Time/sec	RMSE	PSNR	SSIM
1	422724	94.21	0.00517 ±0.00301	21.14819 ±3.41591	0.71965 ±0.08368
2	836484	139.15	0.00289 ±0.00143	25.80587 ±2.90384	0.87870 ±0.03385
4	1663874	213.18	0.00230 ±0.00112	27.89006 ±2.93562	0.91707 ±0.0241
6	2491394	304.02	0.00214 ±0.00104	28.19068 ±3.10928	0.92746 ±0.03523
8	3318914	366.32	**0.00206 ±0.00100**	**28.45171 ±3.06335**	**0.94000 ±0.02676**

Table 4.4: cross all training data-set for various numbers of Fourier layers. Trained over 20 Epoch with dataset size of 10k (8:2) split. The table records the quality of the reconstruction achieved by the Fourier Neural Operator. The means and associated standard deviation are recorded for each one of the metrics: peak-signal-to-noise ratio (PSNR), root-mean-squared error (RMSE) and structural similarity index metric (SSIM). The reconstruction results were tested 2000 images with each image representing a different absorption coefficient distribution μ_a.

As the number of layers increase the network is able to better distinguish discrete boundaries. Which we can visually see below, the improvement is likely due to the network learning to detect 2D curved edges associated with ellipses.

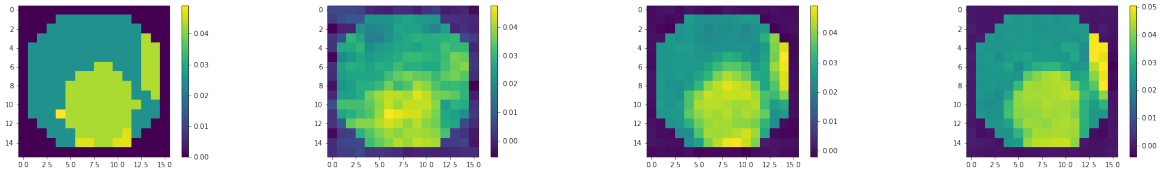

Figure 4.21: Improvement in reconstructions achieved when increasing the number of layers. From left to right: The ground truth, 1 Fourier layer, 2 Fourier layers, 4 Fourier layers and 6 Fourier layers

An unexpected limitation of this network is that the number of modes that exist when a Fourier transform is taken is defined by the size of the input data. This constrain is likely from the nyquist's theorem which states that the 'the sampling frequency must be at least double the highest frequency of the signal', putting an upper most limit on the number of modes that can be used.

By focusing on a single Fourier neural architecture, which in our case is the 8 Fourier layer architecture, we can better understand what the network learns. By investigating the outliers for varying metrics, performance and limitations of the architecture can be established.

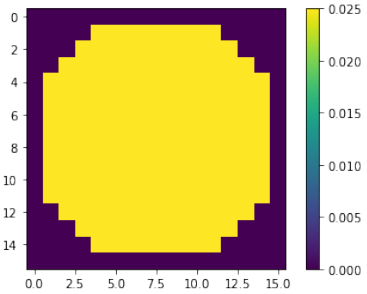

Figure 4.22: Ground truth for FNO-2D PSNR outliers.

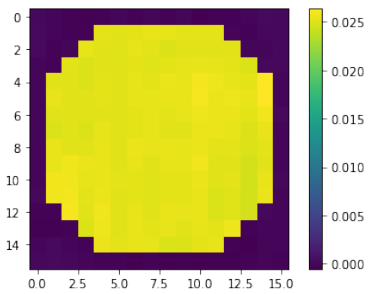

Figure 4.23: Reconstruction PSNR: 37.42793

The a single reconstruction from the network was considered to be a PSNR outlier with a value above 37.36579. The μ_a image was composed of a single ellipse taking up the whole domain with a μ_a coefficient value of 0.25.

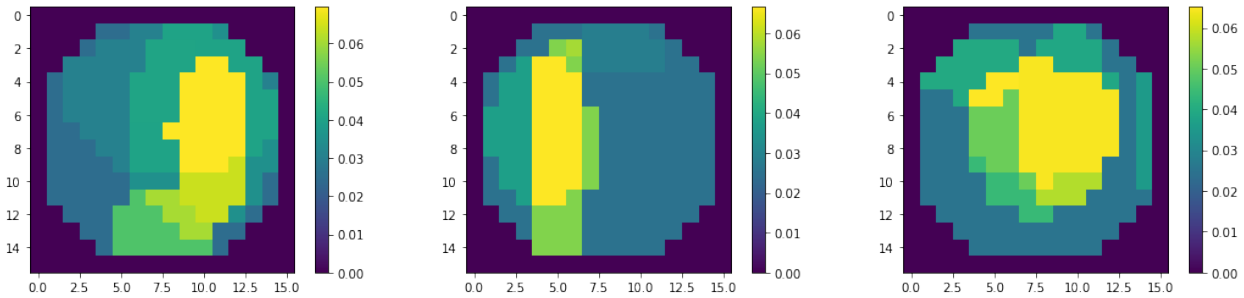

Figure 4.24: Ground truth for FNO RMSE outliers.

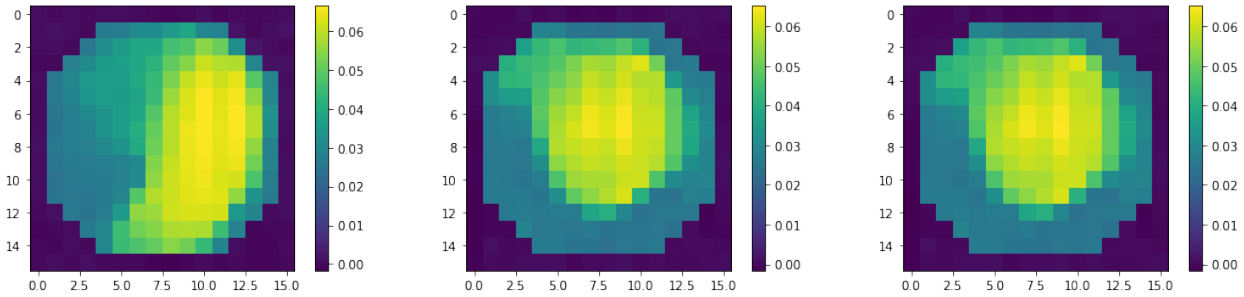

Figure 4.25: Reconstruction RMSE: 0.00600 (left), 0.00650 (centre) and 0.00584 (right).

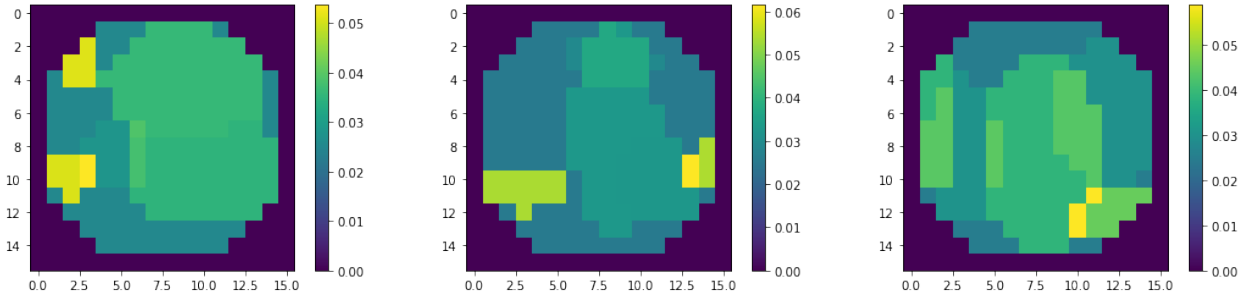

Figure 4.26: Ground truth for FNO SSIM outliers.

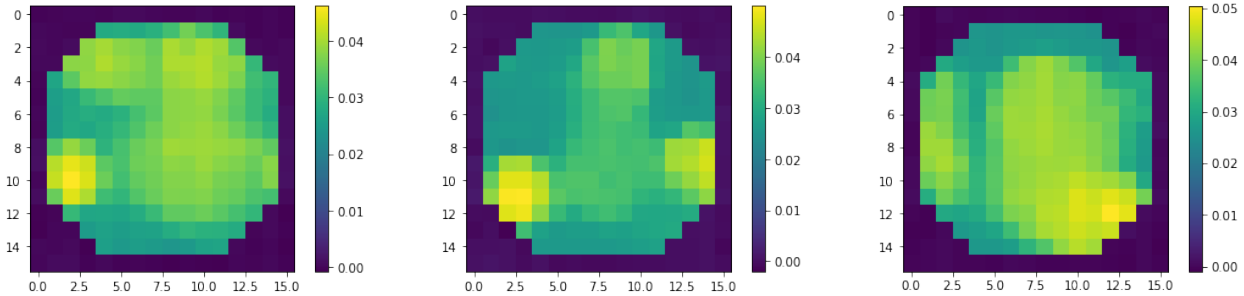

Figure 4.27: Reconstruction SSIM: 0.77182 (left), 0.80732 (centre) and 0.81657 (right).

Notice how the μ_a images which the Fourier neural operator appears to struggle with, are all images composed of multiple overlapping ellipses. If one of those ellipses happens have greater μ_a coefficient value than the other ellipses the network reconstructs the image with a poorer RMSE value. There were three such outliers all with RMSE values above 0.00583. While in total there were 43 images with a significantly poorer SSIM value. All below 0.83621, with the worst being 0.77182. The Fourier neural architecture is capable of learning both local and global variables, though there is a strong preference, similar to the

convolutions neural network, for μ_a images composed of simpler geometries. The network learns both discrete and periodic functions.

This behaviour is further confirmed when we investigated the learned weights of the Fourier neural operator. Once the model has been trained the spectral weights are complex as seen in figure 4.28 and 4.29, and thus have both real and imaginary components.

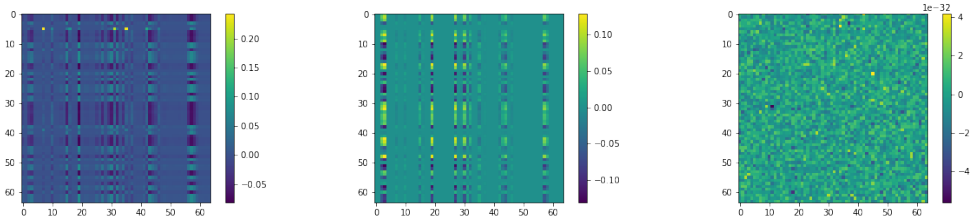

Figure 4.28: Real component of spectral filter: Layer 1, Fourier modes 1 (left), 3 (centre) and 5 (right)

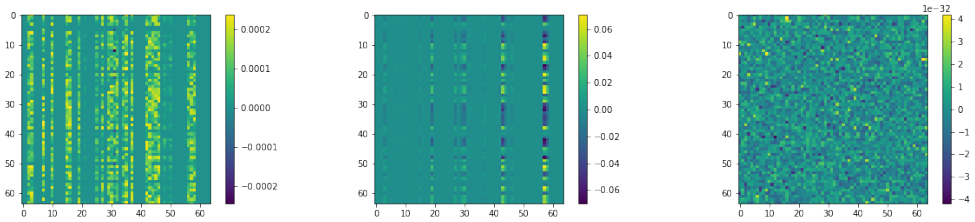

Figure 4.29: Imaginary component of spectral filter: Layer 1, Fourier modes 1 (left), 3 (centre) and 5 (right)

In the figure above the spectral filters are trained in Fourier space, thus have learned to identify periodic spatial behaviour within the measurement data. Filters at higher Fourier modes appear to learn more high frequency components. Spectral filters deeper within the network increasingly learn finer details from the Fourier space, this can be seen within the appendix. Thus increasing the number of Fourier layers improves the networks performance as it is able to learn increasingly finer details. As such the spectral filters are global sinusoidal functions and are better for representing continuous functions.

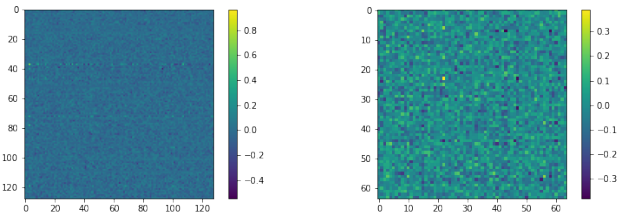

Figure 4.30: A direct comparison of the convolutional filters from two different architectures, the CNN architecture (left) and the FNO architecture (right) — the convolutional filters within the FNO is extracted from the Fourier layer (which is composed of a spectral and convolutional filter).

In contrast the convolution filters within the Fourier layer learns from real space, as seen in figure 4.30. These filters are purely real while the spectral filters are complex. The convolution filters provides the Fourier neural operator with the ability to recognise local details — as is the case with filters within convolutional neural networks.

The spectral filter on its own loses higher frequency modes and works only with periodic boundary conditions. However, the Fourier neural operator as a whole does not have these limitations as the convolutional filter W helps to recover the non-periodic boundaries. (Frequency modes outside of the specified number of Fourier modes are cut-off).

4.5 Comparison and Impact of Noise

Having now established the performance of each one of our networks and variations of each network in ideal conditions, we investigating the impact of noise on the best performing network for each class of architecture. In short we investigated the robustness of the following architectures: multilayered perceptron (single hidden layer, 400 node), convolutional neural network (one fully connected layer, one 5 kernel convolutional layer followed by two 1 kernel convolutional layers) and the Fourier neural operator (8 Fourier layers). Each one of these architectures had the best balance of performances metrics within their class of architectures. (For clarity a single hidden layer is equivalent to three fully connected layers - an input layer, a hidden layer and an output layer).

Each one of these networks were trained and tested on various data-sets with varying

levels of noise. Earlier we had established the performance of models in a noise free environment though within day to day life, measurements are rarely noise free and there is always some level of corruption within our data-set. To simulate such corruption uniform noise of differing noise-levels η is added to the measurements within our data-set. The noise levels represent the amount of noise η within measuring environment. More formally, η represents a uniformly distributed random set of N numbers between the interval $(0,\eta)$, where N here represent size of measurement. Each μ_a distribution is reconstructed from a set of 64 measurements, thus the size of our noise vector is 64 (or an 8×8 matrix).

Noise Level	Itr No	Recon Time /sec	RMSE	PSNR	SSIM
0	22.55 ±3.31	25.89 ±3.87	0.00213 ±0.00108	27.55969 ±3.27086	0.91209 ±0.02914
0.005	23.39 ±6.59	27.74 ±8.21	0.00253 ±0.00095	25.43613 ±2.02679	0.87074 ±0.05766
0.01	20.03 ±5.02	23.66 ±6.13	0.00341 ±0.00083	22.49603 ±1.95159	0.79566 ±0.09943
0.015	16.67 ±4.83	19.20 ±5.54	0.00427 ±0.00073	20.40019 ±2.02651	0.73129 ±0.12302
0.02	13.80 ±5.30	16.01 ±6.18	0.00517 ±0.00097	18.75978 ±2.5377	0.68429 ±0.13391
0.025	10.72 ±4.49	12.17 ±5.11	0.00573 ±0.00092	17.83087 ±2.28276	0.64791 ±0.1335
0.03	9.64 ±4.76	10.92 ±5.40	0.00646 ±0.0011	16.80099 ±2.4754	0.6164 ±0.13929

Table 4.5: Reconstruction metrics using Gauss Newton algorithm. Itr No and Recon Time refer to the number of iterations required and time taken to reconstruct a single μ_a image.

Model Architecture	Noise Level	Training Time /sec	RMSE	PSNR	SSIM
MLP No of Params: 128656	0	32.53	0.00331 ±0.00149	23.65145 ±2.72289	0.81981 ±0.04391
	0.005	32.55	0.00344 ±0.00150	23.26226 ±2.55807	0.81365 ±0.04498
	0.01	30.74	0.0039 ±0.00139	22.36549 ±1.81364	0.80897 ±0.04553
	0.015	30.84	0.0039 ±0.00154	21.72747 ±2.32921	0.78137 ±0.05267
	0.02	31.07	0.00399 ±0.00161	21.61669 ±2.40573	0.76339 ±0.05967
	0.025	31.42	0.00434 ±0.00160	20.79201 ±2.28367	0.74772 ±0.06330
	0.030	33.73	0.00418 ±0.00162	21.06580 ±2.39681	0.74699 ±0.07151
CNN No of Params: 36609	0	63.18	0.00339 ±0.00151	23.50601 ±2.90928	0.82202 ±0.04463
	0.005	61.46	0.00351 ±0.00151	22.88051 ±2.81596	0.81002 ±0.04801
	0.01	64.5	0.00366 ±0.00153	22.60085 ±2.61711	0.8038 ±0.05437
	0.015	61.18	0.00376 ±0.00145	22.30997 ±2.20988	0.78619 ±0.05489
	0.02	65.54	0.00391 ±0.00152	21.79024 ±2.42851	0.77100 ±0.05824
	0.025	65.57	0.00416 ±0.00136	21.57605 ±1.70535	0.76931 ±0.05559
	0.03	63.26	0.00422 ±0.00173	20.64745 ±2.75299	0.75768 ±0.06068
FNO No of Params: 3319044	0	384.85	0.00204 ±0.00109	28.59193 ±3.63352	0.93552 ±0.02888
	0.005	384.64	0.00246 ±0.00116	26.79982 ±2.95597	0.90756 ±0.03700
	0.01	382.61	0.00282 ±0.00133	25.41188 ±3.19985	0.88129 ±0.04813
	0.015	382.66	0.00304 ±0.00154	24.8663 ±3.05732	0.86051 ±0.05702
	0.02	381.79	0.00335 ±0.00198	23.71712 ±3.27411	0.83691 ±0.08080
	0.025	381.10	0.00385 ±0.00285	22.93392 ±3.72971	0.80710 ±0.11521

Table 4.6: Each architecture was trained and tested on data-sets with varying levels of noise.

In table 4.5 we capture the performance of the Gauss-Newton algorithm at differing noise levels, while in Table 4.6 captures the performance of various deep learning architectures. Within this table the deep learning architectures were trained on measurement data-sets with noise levels η, before being tested on data-set with the same noise levels; thus the noise levels within the data did not change between training and testing. In the case where noise levels were low, the Gauss-Newton algorithm had superior performance, only being bested by the Fourier neural operator. Though as noise levels were incrementally increased, every one of the deep learning architectures bested iterative reconstructions achieved by the Guass-Newton algorithms.

Model Architecture	Noise Level	Training Time /sec	Inference Time /sec	RMSE	PSNR	SSIM
MLP No of Params: 128656	0	31.64	1.77	0.00357 ±0.00148	23.01417 ±2.27613	0.79207 ±0.05749
	0.005	31.64	1.77	0.00362 ±0.00148	22.8942 ±2.22816	0.78951 ±0.05819
	0.01	31.64	1.76	0.00374 ±0.00146	22.57381 ±2.05565	0.78239 ±0.06115
	0.015	31.64	1.75	0.00392 ±0.00145	22.14856 ±1.92593	0.77196 ±0.06535
	0.02	31.64	1.84	0.00417 ±0.00142	21.59233 ±1.82472	0.75727 ±0.06937
	0.025	31.64	1.79	0.00447 ±0.00143	21.06573 ±1.79185	0.74128 ±0.07419
	0.03	31.64	1.77	0.00478 ±0.00149	20.58695 ±1.80042	0.72606 ±0.08024
CNN No of Params: 36609	0	62.52	2.17	0.00348 ±0.00152	22.98844 ±2.72778	0.81028 ±0.04544
	0.005	62.52	2.14	0.0035 ±0.00151	22.92204 ±2.68487	0.80812 ±0.04591
	0.01	62.52	2.17	0.00355 ±0.00150	22.7654 ±2.57517	0.80281 ±0.04821
	0.015	62.52	2.19	0.00365 ±0.0015	22.49798 ±2.42817	0.7939 ±0.05122
	0.02	62.52	2.23	0.00378 ±0.00149	22.17693 ±2.31951	0.78338 ±0.05531
	0.025	62.52	2.26	0.00393 ±0.00148	21.81782 ±2.18874	0.77079 ±0.05949
	0.03	62.52	2.16	0.00409 ±0.00148	21.46701 ±2.08407	0.75888 ±0.06573
FNO No of Params: 3319044 Modes: 5	0	400.38	11.51	0.00207 ±0.00106	28.73664 ±3.36533	0.93697 ±0.02843
	0.005	400.38	11.42	0.00284 ±0.00104	25.68212 ±2.13501	0.86939 ±0.05670
	0.01	400.38	11.32	0.00441 ±0.00131	22.17015 ±1.94038	0.76041 ±0.09069
	0.015	400.38	11.60	0.00639 ±0.00234	19.35708 ±2.67351	0.65107 ±0.13171
	0.025	400.38	11.39	0.01096 ±0.0041	15.24684 ±3.32639	0.43803 ±0.19252
	0.03	400.38	11.32	0.01287 ±0.00437	14.21943 ±3.12413	0.36700 ±0.18968

Table 4.7: In many commercial settings it is common to first train the model with ideal noiseless data, before using the trained model in a noisy environment. Thus to determine the various neural networks robustness, we first trained each network with a data-set devoid of noise before providing it with data-sets of varying levels of noise.

In table 4.6 the Fourier neural operator outperforms both the convolutional neural network and the multilayered perceptron, even outperforming the iterative Gauss-Newton method as seen in table 4.5. As long as the Fourier neural operator is trained on measurement data with the same noise profile as the data it is tested on, performance remains stable. Though in the case where the network is trained on data free of noise, only to be tested on noisy data the Fourier neural operators performance dramatically drops. In table 4.7, when the noise levels rise to $\eta = 0.01$, the Fourier neural operator has a lower SSIM value than both the convolutional neural network and the multilayered perceptron.

Clearly, the Fourier neural operator is learning the frequency profile of the additive uniform noise and learning to ignore it. By training the Fourier neural operator on a set of measurements which are free from noise, but introducing noise when testing the architecture, the models performance suffers. It has not been given the opportunity to learn to ignore the noise, which explains the very rapid deterioration in performance, we see in in table

4.7. The Fourier neural operator as an architecture lacks robustness when it comes to noise and in fact clearly needs to be trained on a measurement data with the additive noise, thus simulating the environment which it would be tested in.

Some of these shortcomings can be overcome by discarding some of the higher frequency Fourier modes. By discarding high frequencies, (which is done by changing the number of Fourier modes input into the spectral filter), the architectures performance is reduced but its robustness is improved; as seen in table B.1 within the appendix. By ignoring high spatial frequency components within the measurements, we can reduce the impact of noise. The assumption being that information regarding the μ_a absorption coefficients distribution is more likely to be distributed with a lower periodic spatial frequency within the measurements, while the noise is uniformly distributed. Thus when looking at the measurements, one expects to find proportionally more noise at higher spatial frequencies. (There is more noise than signal at higher frequencies, despite the spatial frequency of noise being uniform, as more of the signal we are interested in exists at lower spatial frequencies).

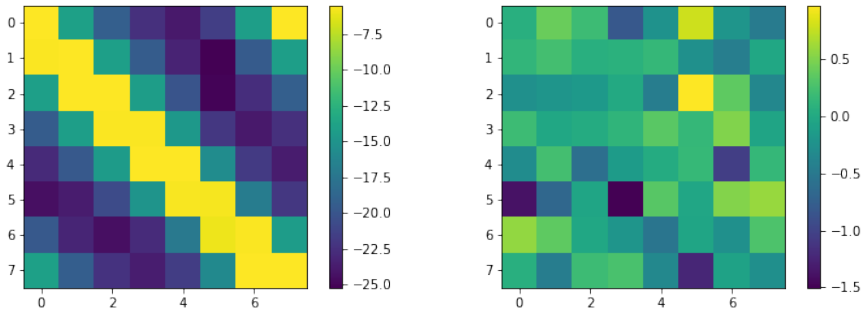

Figure 4.31: Measurements with noise exhibiting periodic behaviour (left) and ground truth removed from measurements with noise, leaving behind the distribution of random uniform noise (right)

In figure 4.31, notice that the measurements, despite corrupted by noise with a noise levels $\eta = 0.030$, shows a periodic behaviour in the form of diagonal streaks. This low spatial frequency characteristic does not exist within the noise. In fact every measurement shows this periodic behaviour. Once we remove the ground truth measurements from our noisy measurements we are left with uniform noise. Clearly this spatial periodic behaviour

provides us with information directly related to our problem and this periodic behaviour is best captured by the lower Fourier modes. Thus as we expect we can improve the robustness of the Fourier neural operator across a range of noise levels by creating an architecture that ignores higher Fourier modes, though it sacrifices performance at low noise levels.

Chapter 5

Conclusion

We have clearly demonstrated the superiority of the Fourier neural operator at solving the inverse diffuse optical tomography problem in comparison to both convolutions architectures, multilayered perceptions and the Gauss-Newton algorithm. To the best of our knowledge the Fourier neural operator has never been applied to the diffuse optical tomography problem and thus we have shown the networks capacity to solve a severely ill-posed partial differential inverse problem. We also show that the Fourier neural operator lacks robustness and is severely impacted by noise, this can be compensated by reducing the number of Fourier modes which the network trains over.

A limitation within the project was the size of our data-set. The time required for a solver to generated 10,000 data points on a local machine was in the order of days, attempting to generate a larger data-set would have required weeks if not months. Attempts were made to make use of the high performance clusters, though this came with a steep learning curve and eventually it was concluded it was better to train models on smaller data-sets and obtain preliminary results. Numerous decisions were made in order to accommodate the slow speed of the forward solver. The synthetic images that were created were limited in size to 16×16, which also put an upper limit on the granularity of the mesh and grid. It makes little sense for the grid in which the image is being mapped to, to have a higher resolution than the

initial image.

By limiting the problem to diffuse optical tomography, we are able to make a very clear claims in regards to the performance of a given architecture. On the other hand it does not give us a definitive answer on how do these architectures perform on partial differential inverse problems as a whole. There are a number of directions I would potentially like to explore going forwards. One of which involves establishing whether performance holds if the imaging modality is changed. Another imaging modality where the forwards operator is modelled by a partial differential equation is electrical impedance tomography; where the forwards operator is modelled by the Poisson equation.[57] This would allow us to establish the generality of which deep learning models are capable of solving partial differential inverse problems. Another noteworthy area of investigation is architectural changes that can be made to the Fourier neural operator as well its performance on a variety of noise profiles. There may be value in determine whether performance improves when the Fourier neural operator is provided with the detector and source positions as a feature.[58] We will explore these possibilities in future works.

Bibliography

[1] Martin Schweiger and Simon R Arridge. "The Toast++ software suite for forward and inverse modeling in optical tomography". In: *Journal of biomedical optics* 19.4 (2014), p. 040801.

[2] Yanna Bai et al. "Deep learning methods for solving linear inverse problems: Research directions and paradigms". In: *Signal Processing* 177 (2020), p. 107729.

[3] Isaac Newton. *De analysi per aequationes numero terminorum infinitas*. 1711.

[4] Ron C Mittelhammer, George G Judge, and Douglas J Miller. *Econometric foundations pack with CD-ROM*. Cambridge University Press, 2000.

[5] Carl Friedrich Gauss. *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*. Vol. 7. FA Perthes, 1877.

[6] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[7] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[8] Geoffrey Hinton et al. "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups". In: *IEEE Signal processing magazine* 29.6 (2012), pp. 82–97.

[9] Terrence J Sejnowski. *The deep learning revolution*. MIT press, 2018.

[10] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5 (1989), pp. 359–366.

[11] Tianping Chen and Hong Chen. "Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems". In: *IEEE Transactions on Neural Networks* 6.4 (1995), pp. 911–917.

[12] Lu Lu et al. "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators". In: *Nature Machine Intelligence* 3.3 (2021), pp. 218–229.

[13] Philip D Wasserman and Tom Schwartz. "Neural networks. II. What are they and why is everybody so interested in them now?" In: *IEEE expert* 3.1 (1988), pp. 10–15.

[14] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[15] Zongyi Li et al. "Fourier neural operator for parametric partial differential equations". In: *arXiv preprint arXiv:2010.08895* (2020).

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[17] Yann LeCun. "The MNIST database of handwritten digits". In: *http://yann. lecun. com/exdb/mnist/* (1998).

[18] Ronan Collobert and Jason Weston. "A unified architecture for natural language processing: Deep neural networks with multitask learning". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 160–167.

[19] Wei Zhang et al. "Shift-invariant pattern recognition neural network and its optical architecture". In: *Proceedings of annual conference of the Japan Society of Applied Physics*. 1988, pp. 2147–2151.

[20] Nikola Kovachki et al. "Neural operator: Learning maps between function spaces". In: *arXiv preprint arXiv:2108.08481* (2021).

[21] Zongyi Li et al. "Neural operator: Graph kernel network for partial differential equations". In: *arXiv preprint arXiv:2003.03485* (2020).

[22] Augustin Cauchy et al. "Méthode générale pour la résolution des systemes d'équations simultanées". In: *Comp. Rend. Sci. Paris* 25.1847 (1847), pp. 536–538.

[23] Jacques Hadamard. *Mémoire sur le problème d'analyse relatif à l'équilibre des plaques élastiques encastrées*. Vol. 33. Imprimerie nationale, 1908.

[24] Paul J Werbos. "Backpropagation through time: what it does and how to do it". In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560.

[25] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[26] Jaejun Yoo et al. "Deep learning diffuse optical tomography". In: *IEEE transactions on medical imaging* 39.4 (2019), pp. 877–887.

[27] Sohail Sabir et al. "Convolutional neural network-based approach to estimate bulk optical properties in diffuse optical tomography". In: *Applied optics* 59.5 (2020), pp. 1461–1470.

[28] Meghdoot Mozumder et al. "A model-based iterative learning approach for diffuse optical tomography". In: *IEEE Transactions on Medical Imaging* 41.5 (2021), pp. 1289–1299.

[29] Yoko Hoshi and Yukio Yamada. "Overview of diffuse optical tomography and its clinical applications". In: *Journal of biomedical optics* 21.9 (2016), p. 091312.

[30] Ronald X Xu and Stephen P Povoski. "Diffuse optical imaging and spectroscopy for cancer". In: *Expert review of medical devices* 4.1 (2007), pp. 83–95.

[31] SA Carp and Q Fang. "Diffuse optical imaging". In: (2014).

[32] David A Boas et al. "Imaging the body with diffuse optical tomography". In: *IEEE signal processing magazine* 18.6 (2001), pp. 57–75.

[33] Pierre Bouguer. *Essai d'optique, sur la gradation de la lumiere*. Claude Jombert, 1729.

[34] Donald F Swinehart. "The beer-lambert law". In: *Journal of chemical education* 39.7 (1962), p. 333.

[35] John M Murkin and M Arango. "Near-infrared spectroscopy as an index of brain and tissue oxygenation". In: *British journal of anaesthesia* 103.suppl_1 (2009), pp. i3–i13.

[36] Rand K Almajidy et al. "A newcomer's guide to functional near infrared spectroscopy experiments". In: *IEEE Reviews in Biomedical Engineering* 13 (2019), pp. 292–308.

[37] Brian W Pogue et al. "Comparison of imaging geometries for diffuse optical tomography of tissue". In: *Optics express* 4.8 (1999), pp. 270–286.

[38] Igor Meglinski. *Biophotonics for medical applications*. Elsevier, 2015.

[39] Subrahmanyan Chandrasekhar. *Radiative Transfer*. Courier Corporation, 1960.

[40] Kenneth M Case. *Linear transport theory*. Addison-Wesley Publishing Company, 1967.

[41] Boris Davison and John Bradbury Sykes. *Neutron transport theory*. Oxford: Clarendon Press, 1957.

[42] Samuel Glasstone and Milton C Edlund. *The elements of nuclear reactor theory*. van Nostrand, 1952.

[43] Turgut Durduran et al. "Diffuse optics for tissue monitoring and tomography". In: *Reports on progress in physics* 73.7 (2010), p. 076701.

[44] Scott Alan Prahl. *Light transport in tissue*. The University of Texas at Austin, 1988.

[45] Wai-Fung Cheong, Scott A Prahl, and Ashley J Welch. "A review of the optical properties of biological tissues". In: *IEEE journal of quantum electronics* 26.12 (1990), pp. 2166–2185.

[46] Lihong V Wang and Hsin-i Wu. *Biomedical optics: principles and imaging*. John Wiley & Sons, 2012.

[47] Niki Munk et al. "Noninvasively measuring the hemodynamic effects of massage on skeletal muscle: a novel hybrid near-infrared diffuse optical instrument". In: *Journal of bodywork and movement therapies* 16.1 (2012), pp. 22–28.

[48] KM Yoo, Feng Liu, and RR Alfano. "When does the diffusion approximation fail to describe photon transport in random media?" In: *Physical review letters* 64.22 (1990), p. 2647.

[49] Adolf Fick. "Ueber diffusion". In: *Annalen der Physik* 170.1 (1855), pp. 59–86.

[50] Simon R Arridge. "Optical tomography in medical imaging". In: *Inverse problems* 15.2 (1999), R41.

[51] Richard C Haskell et al. "Boundary conditions for the diffusion equation in radiative transfer". In: *JOSA A* 11.10 (1994), pp. 2727–2741.

[52] Martin Schweiger, Simon R Arridge, and Ilkka Nissilä. "Gauss–Newton method for image reconstruction in diffuse optical tomography". In: *Physics in Medicine & Biology* 50.10 (2005), p. 2365.

[53] Karan Kumar Pradhan and Snehashish Chakraverty. "Computational structural mechanics". In: *Matthew Deans, California* (2019).

[54] Junuthula Narasimha Reddy. *Introduction to the finite element method*. McGraw-Hill Education, 2019.

[55] Jonas Adler and Ozan Öktem. "Learned primal-dual reconstruction". In: *IEEE transactions on medical imaging* 37.6 (2018), pp. 1322–1332.

[56] Tina Samajdar, Md Quraishi, et al. "Analysis and evaluation of image quality metrics". In: *Information Systems Design and Intelligent Applications*. Springer, 2015, pp. 369–378.

[57] Liliana Borcea. "Electrical impedance tomography". In: *Inverse problems* 18.6 (2002), R99.

[58] Zongyi Li et al. "Fourier Neural Operator with Learned Deformations for PDEs on General Geometries". In: *arXiv preprint arXiv:2207.05209* (2022).

Appendices

Appendix A

Definitions

Definition A.0.1 (Jacobian)**.** Suppose a function $\boldsymbol{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$, is defined for all $\mathbf{x} = (x_1, x_2, ..., x_n)$ by

$$\begin{aligned}\boldsymbol{f}(\mathbf{x}) &= \boldsymbol{f}(x_1, x_2, \ldots, x_n) \\ &= (f_1(x_1, x_2, \ldots, x_n), , \ldots, f_m(x_1, x_2, \ldots, x_n)) \\ &= (f_1(\mathbf{x}), , \ldots, f_m(\mathbf{x}))\end{aligned}$$

Thus the Jacobian is

$$D = \begin{bmatrix} \dfrac{\partial \boldsymbol{f}(\mathbf{x})}{\partial x_1} & \cdots & \dfrac{\partial \boldsymbol{f}(\mathbf{x})}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla^T f_1(\mathbf{x}) \\ \vdots \\ \nabla^T f_m(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f_1(\mathbf{x})}{\partial x_1} & \cdots & \dfrac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_m(\mathbf{x})}{\partial x_1} & \cdots & \dfrac{\partial f_m(\mathbf{x})}{\partial x_n} \end{bmatrix}$$

Definition A.0.2 (Laplace's Spherical Harmonics)**.** Any well-behaved function $f(\phi, \theta)$ can be expanded in terms of complex spherical harmonics, such that

$$f(\phi, \theta) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} f_{m,l} Y_{m,l}(\phi, \theta)$$

for some choice of $f_{m,l}$, where the spherical harmonics $Y_{m,l}$ form a complete orthonormal system, such that

$$\int Y_{m,l}(\theta, \phi) Y_{m',l'}(\theta, \phi) d\Omega = \delta_{ll'}\delta_{mm'} \quad \text{and} \quad Y_{m,-l} = (-1)^m Y*_{m,l}$$

where $d\Omega = \sin\theta d\theta d\phi$ is the differential solid angle in spherical coordinates.

Definition A.0.3 (Associated Legendre Polynomials)**.** The spherical harmonics arise from solving Laplace's equation

$$\nabla^2 \psi = 0$$

in spherical coordinates. We can make the substitution $\psi(r, \theta, \phi) \equiv R(r)Y(\theta, \phi)$, thus making the equation now separable. After making the appropriate substitutions, separation of variables and re-arrangements.

The first equation which we end up with after making the appropriate substitutions is

$$\frac{d^2}{d\phi^2}\Phi_m(\phi) + m^2 \Phi_m(\phi) = 0$$

where $\Phi_m(\phi) = \exp(im\phi)$ is the solution to the expression with m being a separation constant.

After further making use of properties of Euler differential equations the second equation

takes the form of an associated Legendre differential equation and thus can be rewritten as

$$(1-x^2)\frac{d^2y}{dx^2} - 2x\frac{dy}{dx} + \Big[l(l+1) - \frac{m^2}{1-x^2}\Big]y = 0$$

where $x = \cos\theta$ and $y = P_{m,l}$ of degree l and order m is the solution to the equation. Note that l is another separation constant subject to the constraint $l \geq 0$ and $|m| \leq l$.

Now we can finally show that the the normalised spherical harmonics are

$$Y_{l,m}(\theta,\phi) = (-1)^m \sqrt{\frac{(2l+1)}{4\pi}\frac{(l-m)!}{(l+m)!}}\, P_{m,l}(\cos\theta)e^{im\phi},$$

which form a complete orthogonal set of functions. The spherical harmonics can be used to decompose functions on the sphere.

$$P_{m,l}(\alpha) = \frac{(1-\alpha^2)^{m/2}}{2^l l!}\frac{d^{l+m}}{d\alpha^{l+m}}(\alpha^2-1)^l$$

Definition A.0.4 (Box Plots and Outliers)**.** Outliers by definition are considered anything outside of a certain range of the median of a data-set. Thus the definition of an outlier we use within this thesis is

$$\text{Outlier} < Q_1 - 1.5 \times \text{IQR} \quad \text{and} \quad Q_3 + 1.5 \times \text{IQR} < \text{Outlier} \tag{A.1}$$

where the inter-quartile range ($\text{IQR} = Q_3 - Q_1$) is the difference between the upper quartile (Q_3) or 75th percentile and the lower quartile (Q_1) or 25th percentile. The interquartile range (IQR) is a measure of statistical dispersion and may also be referred to as the midspread.

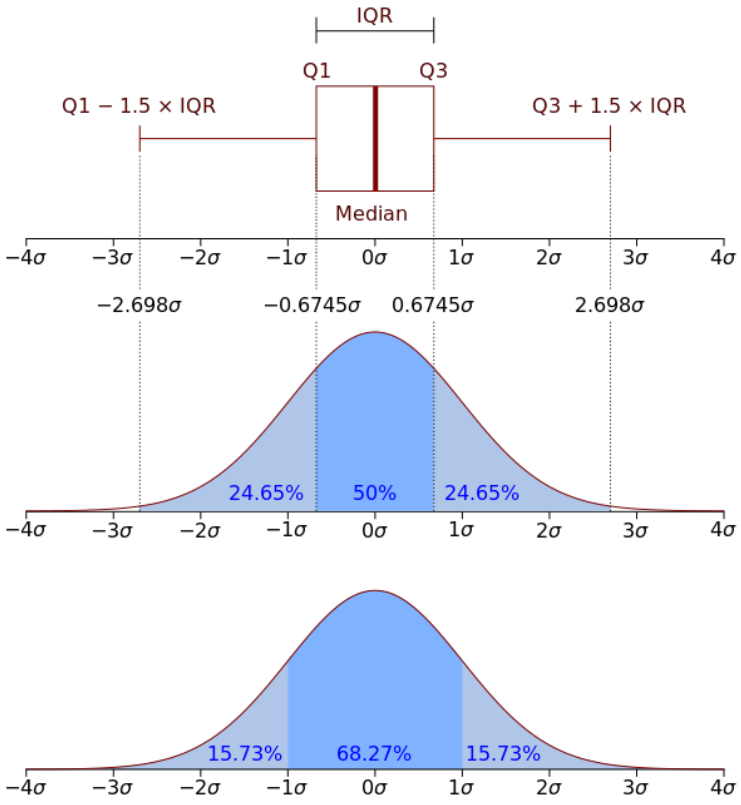

Figure A.1: How to interpret box-plots in the case of a distributed data-set

To provide the reader a quick reminder of how box-plots are related to probability density functions it is worthwhile looking at figure A.1. Box plots have been around since the 1970s and were developed by John Tukey. It shows a distribution summary of a data-set in a small amount of space. Box-plots provide a visual representation of the skew of a data-set and can be used to determine the standard deviation.

Appendix B

Fourier Neural Operator Results

B.1 Changing the Number of Fourier Layers: Reconstructions

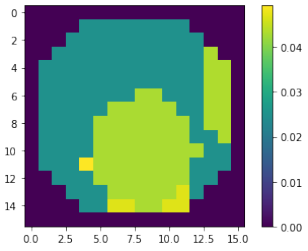

Figure B.1: Ground truth

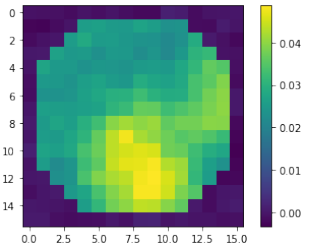

Figure B.2: Reconstruction from FNO with 1 Fourier layer

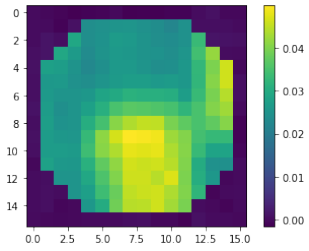

Figure B.3: Reconstruction from FNO with 2 Fourier layer

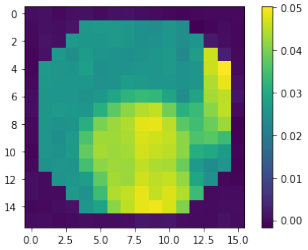

Figure B.4: Reconstruction from FNO with 4 Fourier layer

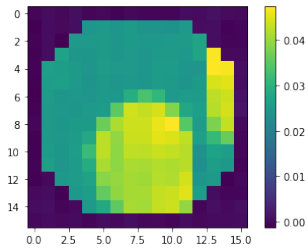

Figure B.5: Reconstruction from FNO with 6 Fourier layer

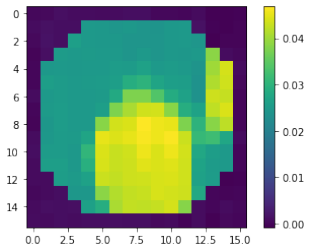

Figure B.6: Reconstruction from FNO with 8 Fourier layer

B.2 Changing the Number of Fourier Layers: Box Plots for Quality Metrics

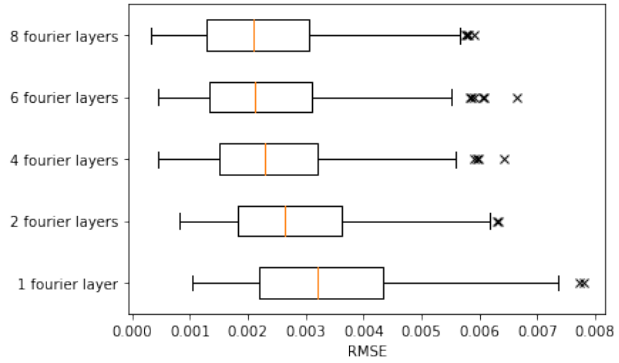

Figure B.7: Boxplot for SSIM

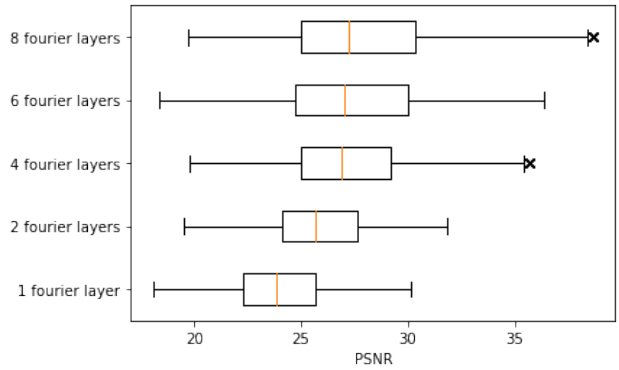

Figure B.8: Boxplot for PSNR

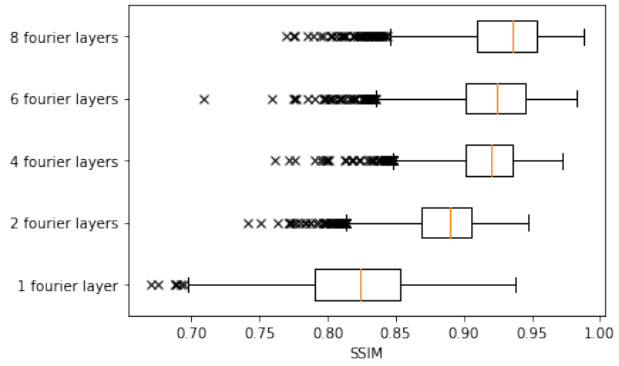

Figure B.9: Boxplot for SSIM

B.3 Trained Fourier Neural Operator Filters

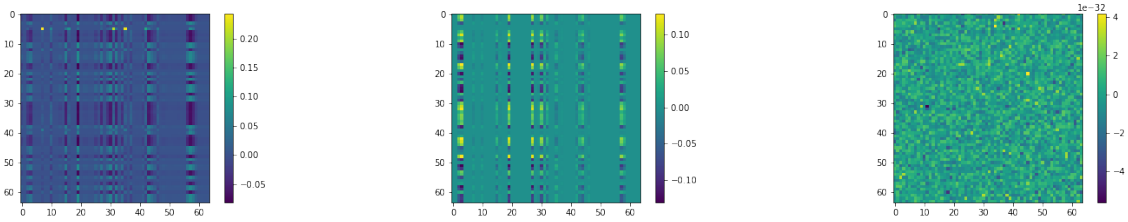

Figure B.10: Real component of spectral filter: Layer 1, Fourier modes 1 (left), 3 (centre) and 5 (right)

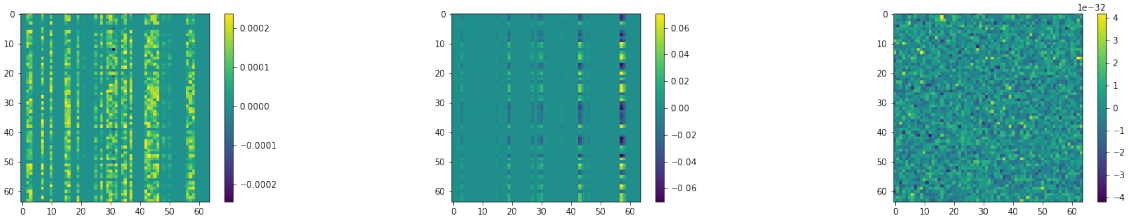

Figure B.11: Imaginary component of spectral filter: Layer 1, Fourier modes 1 (left), 3 (centre) and 5 (right)

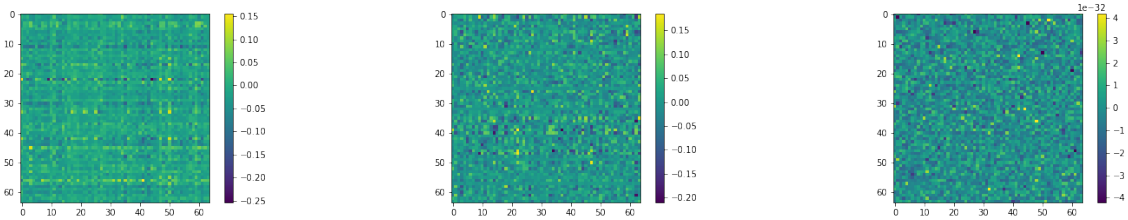

Figure B.12: Real component of spectral filter: Layer 4, Fourier modes 1 (left), 3 (centre) and 5 (right)

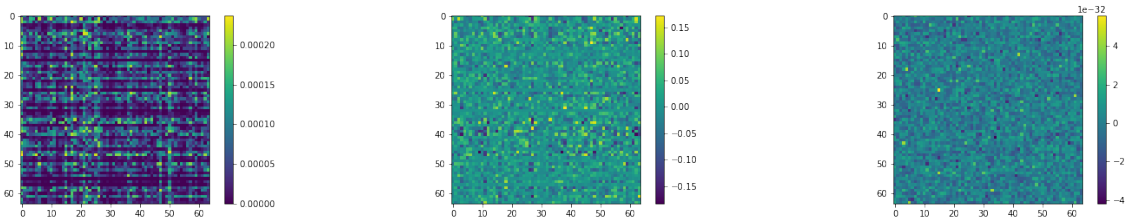

Figure B.13: Imaginary component of spectral filter: Layer 4, Fourier modes 1 (left), 3 (centre) and 5 (right)

B.4 Effect of Changing the Number of Fourier Modes

Model Architecture	Noise Level	Training Time /sec	Inference Time /sec	RMSE	PSNR	SSIM
FNO No of Params: 42244 Mode: 0	0	132.12	3.58	0.01569 ±0.00329	1.65957 ±1.09758	0.04556 ±0.04349
	0.005	132.12	3.51	0.01569 ±0.00329	1.61911 ±1.10472	0.04486 ±0.04332
	0.01	132.12	3.40	0.0157 ±0.00328	1.61168 ±1.11309	0.04433 ±0.04348
	0.015	132.12	3.39	0.01571 ±0.00327	1.65528 ±1.13208	0.04376 ±0.04428
	0.02	132.12	3.36	0.01573 ±0.00327	1.79465 ±1.1511	0.04291 ±0.04446
	0.025	132.12	3.37	0.01576 ±0.00326	1.9085 ±1.1323	0.04114 ±0.04518
	0.03	132.12	3.37	0.01578 ±0.00326	2.09739 ±1.1584	0.04145 ±0.04561
FNO No of Params: 173316 Mode:1	0	383.07	10.89	0.00736 ±0.00262	16.72027 ±1.9667	0.6029 ±0.08814
	0.005	383.07	10.75	0.00771 ±0.0027	16.30726 ±2.04615	0.58523 ±0.09426
	0.01	383.07	10.50	0.00882 ±0.0034	15.18993 ±2.47944	0.52946 ±0.13055
	0.015	383.07	10.64	0.01013 ±0.00396	14.04471 ±2.9006	0.46204 ±0.16494
	0.02	383.07	10.74	0.01176 ±0.00452	12.98616 ±3.07622	0.38327 ±0.19345
	0.025	383.07	10.56	0.0133 ±0.00496	12.23868 ±2.95139	0.31377 ±0.20100
	0.03	383.07	11.49	0.01453 ±0.00501	11.83973 ±2.97923	0.26143 ±0.21023
FNO No of Params: 566532 Modes: 2	0	397.79	11.52	0.00256 ±0.00124	26.83493 ±3.22949	0.89900 ±0.03301
	0.005	397.79	11.35	0.00278 ±0.00124	25.98585 ±2.67438	0.87836 ±0.03766
	0.01	397.79	11.18	0.00343 ±0.00173	24.30222 ±2.43265	0.82943 ±0.07297
	0.015	397.79	11.22	0.00449 ±0.00256	22.32339 ±2.78042	0.76234 ±0.12014
	0.02	397.79	12.44	0.00615 ±0.00395	20.26078 ±3.62921	0.67447 ±0.17926
	0.025	397.79	11.45	0.00818 ±0.00515	18.29052 ±4.14601	0.57754 ±0.22602
	0.03	397.79	11.76	0.00987 ±0.00575	17.01289 ±4.15862	0.50389 ±0.24599
FNO No of Params: 1221892 Modes: 3	0	414.83	11.88	0.00233 ±0.00115	27.6163 ±3.27839	0.91506 ±0.03146
	0.005	414.83	11.83	0.00274 ±0.00114	26.04172 ±2.5209	0.87968 ±0.04281
	0.01	414.83	12.12	0.00372 ±0.0013	23.57423 ±2.01935	0.80801 ±0.07523
	0.015	414.83	11.85	0.00505 ±0.00181	21.31272 ±2.22453	0.72735 ±0.10452
	0.02	414.83	11.74	0.00675 ±0.00282	19.23027 ±2.89714	0.63592 ±0.14584
	0.025	414.83	11.79	0.00874 ±0.00393	17.47087 ±3.3345	0.53956 ±0.18453
	0.03	414.83	11.95	0.0105 ±0.00444	16.20879 ±3.32487	0.46188 ±0.20342
FNO No of Params: 2139396 Modes: 4	0	405.29	11.52	0.00204 ±0.0011	28.87063 ±3.74675	0.93668 ±0.02862
	0.005	405.29	11.61	0.0027 ±0.00111	26.04369 ±2.33727	0.87481 ±0.05472
	0.01	405.29	11.12	0.00407 ±0.00134	22.7394 ±1.99151	0.77657 ±0.08981
	0.015	405.29	11.22	0.00591 ±0.00232	19.91923 ±2.73787	0.67372 ±0.13005
	0.02	405.29	11.28	0.00804 ±0.00339	17.6823 ±3.28212	0.56755 ±0.16846
	0.025	405.29	11.31	0.01029 ±0.00425	15.96082 ±3.43883	0.46524 ±0.19728
	0.03	405.29	12.06	0.01223 ±0.00474	14.96554 ±3.29033	0.39047 ±0.20344
FNO No of Params: 3319044 Modes: 5	0	400.38	11.51	0.00207 ±0.00106	28.73664 ±3.36533	0.93697 ±0.02843
	0.005	400.38	11.42	0.00284 ±0.00104	25.68212 ±2.13501	0.86939 ±0.05670
	0.01	400.38	11.32	0.00441 ±0.00131	22.17015 ±1.94038	0.76041 ±0.09069
	0.015	400.38	11.60	0.00639 ±0.00234	19.35708 ±2.67351	0.65107 ±0.13171
	0.025	400.38	11.39	0.01096 ±0.0041	15.24684 ±3.32639	0.43803 ±0.19252
	0.03	400.38	11.32	0.01287 ±0.00437	14.21943 ±3.12413	0.36700 ±0.18968

Table B.1: Each Fourier layer requires us to specify the number of Fourier modes to keep (discarding the remaining). We can thus determine the impact impact discarding Fourier modes has on the networks ability to learn and reconstruct.